Fast Modeling of Radiation and Conduction Heat Transfer and application example

Boutros Ghannam

ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY

MINES ParisTech

Ecole doctorale n° 432 : Sciences des Métiers de l'Ingénieur

Doctorat ParisTech

T H È S E

pour obtenir le grade de docteur délivré par

l'École nationale supérieure des mines de Paris

Spécialité "Energétique"

présentée et soutenue publiquement par

Boutros GHANNAM

le 19 Octobre 2012

Modélisation Ultra-rapide des Transferts de Chaleur par Rayonnement et par Conduction et exemple d'application

Fast Modeling of Radiation and Conduction Heat Transfer and application example

Directeur de thèse : **Denis CLODIC**
Co-encadrement de la thèse : **Maroun NEMER**

Jury

M. John R. HOWELL, Prof., Mechanical Eng. Department, University of Texas at Austin	Rapporteur
M. Olivier GICQUEL, Prof., Laboratoire EM2C, Ecole Centrale Paris	Rapporteur
M. Ludovic FERRAND, Dr., Technology officer, CMI Greenline Europe	Examinateur
M. Walter YUEN, Prof., Academic Development, The Hong Kong Polytechnic University	Examinateur
M. Denis CLODIC, Directeur de recherche émérite, Mines ParisTech	Examinateur
M. Maroun NEMER, Dr., Centre Energétique et Procédés, Mines ParisTech	Examinateur

MINES ParisTech
Centre Efficacité énergétique des Systèmes
60 bd Saint-Michel 75272 PARIS Cedex 06

T H E S E

ACKNOWLEDGMENTS

This work was carried out at the “Center for Energy and Processes – Paris” (CEP) of the “École Nationale Supérieure des Mines de Paris” (ENSMP) in France.

I first thank my supervisor Pr. Denis Clodic for hosting me in his team and advising me throughout this work. I particularly thank him for his trust and his invaluable always just in time intervention and support.

I thank Dr. Maroun NEMER for his support and guidance. His integral view on research and continuous enthusiasm for always getting better results has pushed this work forward more than I have expected.

I thank Dr. Khalil El KHOURY for his guidance and his great effort for revising and verifying the work throughout the thesis. I also particularly thank him for his enthusiasm and encouragement that always helped me to keep going forward.

I thank Pr. Walter W. Yuen for his support and effort for reviewing a part of this work. I owe him a particular acknowledgment for his work that is the basis for this thesis project.

The work presented in this dissertation was supported and partially funded by CMI INDUSTRY Thermline. I particularly thank Dr. Ludovic Ferrand and Mr. Jean-Christophe MITAIS for all the interest they have shown in this work and also for their valuable support and remarks.

I would like to express my sincere appreciation and gratitude to Anne-Marie Bonnet for her professional support and personal involvement and encouragement throughout this work. I owe her special thanks for her great effort in proofreading my work and making sure that everything goes right.

I also thank Rocio VALDEZ, Marie-Astrid KRAMES, Philomène ANGELOSANTO, and Joëlle ANDRIANARIJAONA for their professional support.

Finally, I thank my colleagues and coworkers in the CEP for their friendship, support and valuable discussions.

I dedicate this work to my parents Dunia and Georges, to my brothers Boulos, Andros, and Angelo and to my girlfriend Abeer. I particularly thank them for their support and encouragement and for enduring my absence during this work.

CONTENTS

INTRODUCTION ***1***

CHAPTER 1 DIRECT EXCHANGE FACTORS COMPUTATION METHOD ***9***

Nomenclature **Erreur ! Signet non défini.**

1.1 Introduction

1.2 Multiple Absorption Coefficient Zonal Method (MACZM)

1.2.1 Concept of Generic Exchange Factors (GEF) and Superposition

1.2.2 Mean Beam Length (MBL) and Artificial Neural Network Correlations

1.3 Modray Tool Description Erreur ! Signet non défini.

1.3.1 Calculation of Direct Exchange Factors sisj with the Flux Planes Method

1.3.2 Calculation of Surface-Volume and Volume-Volume Direct Exchange Factors using Emery Et Al. Relations

1.3.3 Calculation of Total Exchange Factors using the Plating Algorithm

1.4 Elementary Operations Validation Erreur ! Signet non défini.

1.5 Test Case Description Erreur ! Signet non défini.

1.5.1 Furnace Description **Erreur ! Signet non défini.**

1.5.2 Modeling Approach **Erreur ! Signet non défini.**

1.5.3 Test Procedure **Erreur ! Signet non défini.**

1.6 Calculation of Radiative Exchange Factors Erreur ! Signet non défini.

1.6.1 Calculation of the Direct Exchange Factors in the Furnace using MACZM **Erreur ! Signet non défini.**

1.6.2 Calculation of the Direct Exchange Factors using Modray **Erreur ! Signet non défini.**

1.6.3 Comparison of Results And Discussion **Erreur ! Signet non défini.**

1.7 Experimental Validation Erreur ! Signet non défini.

1.8 Conclusions Erreur ! Signet non défini.

References Erreur ! Signet non défini.

CHAPTER 2 IMPLEMENTATION AND PARALLELIZATION OF MACZM ***11***

Nomenclature **Erreur ! Signet non défini.**

2.1 Introduction Erreur ! Signet non défini.

2.2 Multiple Absorption Coefficient Zonal Method (MACZM) And Algorithm Erreur ! Signet non défini.

2.3 Grid Generation And Representation Of The Objects In The Discrete Space Erreur ! Signet non défini.

2.4 Mean Absorption Coefficient Computation By Ray Tracing Erreur ! Signet non défini.

2.4.1 Connectivity of The Discrete Line **Erreur ! Signet non défini.**

2.4.2 Span by Span Algorithm **Erreur ! Signet non défini.**

2.4.3 The Tripod 6-Line Algorithm **Erreur ! Signet non défini.**

2.4.4 The Parametric 6-Line Algorithm **Erreur ! Signet non défini.**

2.4.5 Comparison of the Ray Traversal and Mean Absorption Coefficient Computation Algorithms **Erreur ! Signet non défini.**

2.5 Artificial Neural Network And GEF Superposition Erreur ! Signet non défini.

2.6 GPU Cuda Programming Model Overview Erreur ! Signet non défini.

2.7 Parallel Implementation of MACZM in CUDA Erreur ! Signet non défini.

2.7.1 Cuda Parallel Kernel **Erreur ! Signet non défini.**

2.7.2 Computation of The Mean Absorption Coefficient **Erreur ! Signet non défini.**

2.7.3 Artificial Neural Networks **Erreur ! Signet non défini.**

2.8 Application: Simulation of a Steel Reheating Furnace Erreur ! Signet non défini.

2.8.1 Description of The Steel Reheating Furnace and the Simplified Furnace **Erreur ! Signet non défini.**

2.8.2 Simulation in MODRAY **Erreur ! Signet non défini.**

2.8.3 Simulation in MACZM **Erreur ! Signet non défini.**

2.8.4 Relative Error **Erreur ! Signet non défini.**

2.8.5 Discussions and MACZM Multi-Grid **Erreur ! Signet non défini.**

2.9 Conclusions Erreur ! Signet non défini.

References Erreur ! Signet non défini.

CHAPTER 3 TOTAL EXCHANGE FACTORS COMPUTATION BY NRPA 13

Nomenclature **Erreur ! Signet non défini.**

3.1 Introduction Erreur ! Signet non défini.

3.2 Plating Algorithm Mathematical Formulation Erreur ! Signet non défini.

3.2.1 Matrix Representation And Parallel Execution Of The PA **Erreur ! Signet non défini.**

3.3 Formulation of the Non-Recursive Plating Algorithm (NRPA) Erreur ! Signet non défini.

3.3.1 Identification of the Non-Recursive Plating Algorithm (NRPA) Equation **Erreur ! Signet non défini.**

3.3.2 Nrpa Equation for an Enclosure With 3 Surfaces **Erreur ! Signet non défini.**

3.3.3 Nrpa Equation for an Enclosure With N Surfaces **Erreur ! Signet non défini.**

3.3.4 Demonstration of the Non-Recursive Plating Algorithm (NRPA) Equation By Recurrence**Erreur ! Signet non défini.**

3.3.5 Computation of the $skskk-1$ Exchange Areas **Erreur ! Signet non défini.**

3.3.6 Order of the Non-Recursive Plating Algorithm **Erreur ! Signet non défini.**

3.4 Matrix Form of the NRPA Erreur ! Signet non défini.

3.4.1 NRPA of Order 2 **Erreur ! Signet non défini.**

3.4.2 NRPA of Order M > 2 **Erreur ! Signet non défini.**

3.5 Reducing The Computational Complexity of the Nrpa Via Matrix Multiplication Erreur ! Signet non défini.

3.5.1 Overview of Matrix Multiplication Optimal Algorithms **Erreur ! Signet non défini.**

3.5.2 Computational Complexity of the NRPA **Erreur ! Signet non défini.**

3.6 Testing The Error of Non-Recursive Plating Algorithm Erreur ! Signet non défini.

3.6.1 Enclosure Description **Erreur ! Signet non défini.**

3.6.2 Computation of Direct and Total Exchange Areas **Erreur ! Signet non défini.**

3.6.3 Error Analysis and Discussions **Erreur ! Signet non défini.**

3.7 Implementation and Computation Time Comparison of the Pa And The Nrpa Erreur ! Signet non défini.

3.7.1 Multi-Core Cpu Implementation of the Pa **Erreur ! Signet non défini.**
3.7.2 Gpu Cuda Parallelization of the Pa **Erreur ! Signet non défini.**
3.7.3 Cpu and Gpu Implementation of the Nrpa **Erreur ! Signet non défini.**
3.7.4 Computation Time Comparison **Erreur ! Signet non défini.**

3.8 Conclusions Erreur ! Signet non défini.

References Erreur ! Signet non défini.

CHAPTER 4 3D HEAT DIFFUSION COMPUTATION 15

Nomenclature **Erreur ! Signet non défini.**

4.1 Introduction Erreur ! Signet non défini.

4.2 Discrete Approximation Of The 3d Heat Diffusion Equation By Finite Differences Erreur ! Signet non défini.

4.2.1 Three-Dimensional Heat Diffusion Equation **Erreur ! Signet non défini.**
4.2.2 Discretization of The Heat Diffusion PDE **Erreur ! Signet non défini.**
4.2.3 Simple Explicit Method **Erreur ! Signet non défini.**
4.2.4 Simple Implicit Method **Erreur ! Signet non défini.**
4.2.5 Crank-Nicolson Method **Erreur ! Signet non défini.**

4.3 Finite Differences Split Methods Applied to the 3d Heat Diffusion PDE Erreur ! Signet non défini.

4.3.1 Locally One-Dimensional (LOD) Method **Erreur ! Signet non défini.**
4.3.3 Douglas-Gunn Alternating Direction Implicit (ADI) Method **Erreur ! Signet non défini.**
4.3.4 Alternating Direction Explicit (ADE) Method **Erreur ! Signet non défini.**

4.4 Highly Efficient Tridiagonal Linear System Solvers Erreur ! Signet non défini.

4.4.1 Thomas Algorithm **Erreur ! Signet non défini.**
4.4.2 Cyclic Reduction (CR) Algorithm **Erreur ! Signet non défini.**
4.4.3 Parallel Cyclic Reduction (PCR) Algorithm **Erreur ! Signet non défini.**

4.5 Gpu Cuda Programming Model Overview Erreur ! Signet non défini.

4.5.1 Maximizing The Number of Parallel Threads **Erreur ! Signet non défini.**
4.5.2 Memory Coalescing **Erreur ! Signet non défini.**

4.6 Parallelization of the Split Methods and Implementation in CUDA Erreur ! Signet non défini.

4.6.1 Utmost Parallelisation Implementation **Erreur ! Signet non défini.**
4.6.2 Stencil Implementation **Erreur ! Signet non défini.**
4.6.3 Performance and Comparison **Erreur ! Signet non défini.**

4.7 Highly Efficient GPU Implementation of the LOD Method Using Optimal Solvers and Parallelization Schemes Erreur ! Signet non défini.

4.8 Analytical Validation if the Split Methods 111

4.8.1 Setting the Problem 111
4.8.2 Analytical Solution 111

4.9 Analysis By Example of the Accuracy of the Split Methods in Order to Time and Space Mesh Size 113
4.9.1 Numerical Simulation 113

4.10 Conclusions Erreur ! Signet non défini.

References Erreur ! Signet non défini.

CHAPTER 5 3D ULTRA-RAPID SIMULATION OF STEEL REHEATING FURNACE, OPENING FOR INLINE CONTROL 117

5.1 Introduction 117

5.2 3d Numerical Computation Model for a Steel Reheating Furnace and Performance Evaluation 120
5.2.1 Computation of the Direct Radiative Exchange Factors 121
5.2.2 Computation of the Total Radiative Exchange Factors 122
5.2.3 Computation of temperature Profiles in the Slabs 123
5.2.4 Dynamic Simulation and Performance Evaluation 124

5.3 Application: Thermal Analysis of a Steel Reheating Furnace 125
5.3.1 Thermal Analysis Model 125
5.3.2 Discretization 126
5.3.3 Radiation Heat Exchange in the Furnace 126
5.3.4 Thermal Model of the Walls and the Rails 126
5.3.5 Energy Conservation and Gas Temperature 127

5.4 Optimization of Rails Positions and Heating Temperature Profiles 127
5.4.1 Thermal Model Inputs 127
5.4.2 Black Points Analysis and Rails Positioning 127
5.4.3 Heating Temperature Profile Set Point 130

5.5 Conclusions and Future Work 131

Conclusions and Perspectives 133

APPENDIX A GPU CUDA PROGRAMMING 137

A.1 Introduction 138

A.2 Cuda Program Structure 139

A.3 Cuda Memories 140

A.4 Cuda Threads 141

A.5 Performance Considerations 142

APPENDIX B RESUME DE LA THESE EN FRANCAIS 143

B.1 Calcul des Facteurs de Transferts Radiatifs Directs par MACZM 144

B.2 Implémentation et Parallélisation de MACZM 152

B.3 Calcul Des Facteurs de Transferts Radiatifs totaux 162

B.4 Modélisation de La Diffusion 3D par les méthodes des différences finies 169

B.5 Application à la Simulation Ultra-Rapide d'un Four de Réchauffage Sidérurgique 182

Introduction

Massively Parallel Programming and Fast Computing

For more than two decades since their beginning, Central Processing Units (CPUs) drove rapid performance increases and cost reductions in computer applications. Hardware advances allowed the same programs to run faster with each new CPU generation. This also allowed application software to have better user interfaces and provide more functionality. On the other hand, parallel programs used to run more usually on data-center servers or departmental clusters. As a result, only a few elite applications funded by governments and large corporations have been successfully developed on these parallel computing systems. Although parallel programming was less accessible and had limited market, sequential programming is more naturally understood by human mind.

Since 2003, the increase in single CPU performance has slowed down due to energy consumption and heat dissipation issues that limited the increase of the clock frequency and the work that can be achieved in each CPU clock period. This has enhanced the appearance of multi-core CPUs that begun with two-core processors, that roughly doubles with each new CPU generation. On the other hand, the programmability of Graphical Processing Units (GPUs) has begun to increase. GPUs are many-core processors that were first designed for the video game industry, which exerts tremendous economic pressure for the ability to perform a massive number of floating-point calculations per video frame in advanced games. Figure 1 shows a comparison between CPUs and GPUs theoretical floating-point throughput from 2001 to 2009. In 2009, GPU theoretical floating-point throughput had become about ten times higher than CPU. In the same way as multi-core CPUs, the number of GPU cores tends to double with each new generation and have got to more than 1500 cores per single GPU chip in 2012. As a result, GPUs provides now Terra Floating-Point operations per second (TFLOPS) on Desktop computers and Peta Floating-Point operations per second (PFLOPS) on Clusters.

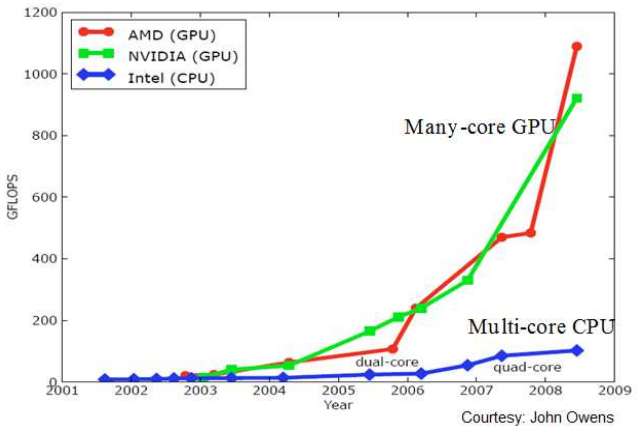


Figure 1: CPUs and GPUs theoretical floating-point throughput from 2001 to 2009.

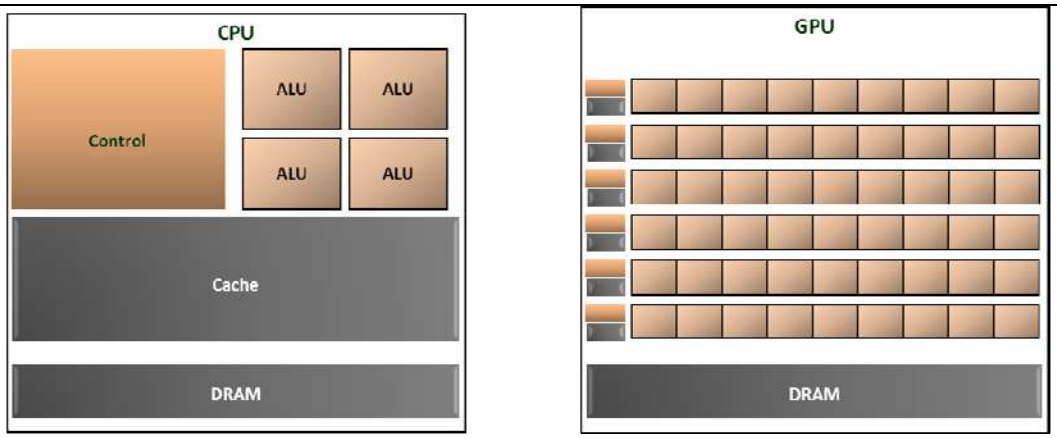


Figure 2: CPU and GPU design comparison

The large performance gap between multi-core CPUs and many-core GPUs is due to the difference in their fundamental design philosophy. CPUs are optimized for sequential run. As shown in Figure 2, they have a large cache memory in order to reduce memory accesses latency. In addition, they are equipped with sophisticated control logic that allows the instructions of a single thread to execute in parallel or out of their sequential order while maintaining the appearance of sequential execution. Nevertheless, the performance is not due to control or cache memory, but to the power of the Arithmetic-Logic Unit (ALU). On the other hand, GPUs are designed to have small cache memory and less complex logic unit, but a very large number of smaller cores. GPU cores are designed to execute in parallel while their execution scheduling is optimized by hardware function. The smaller GPU cache memory is covered by higher memory bandwidth to its DRAM memory due to a lighter design of this one. As a result, the high number of GPU cores is able to deliver high throughput, with lower cost and less energy.

GPUs were first developed for game industry, and thus their programmability was very limited. Even with the low programmability, successful programs were implemented for GPU execution for non-graphical applications, using the difficult graphics APIs. This was known as General Purpose programming using a Graphics Processor Unit (GP-GPU). Thus, only a very few highly skilled programmers were able to develop general purpose applications to run on the GPU. Everything changed with the release of CUDA by NVIDIA in 2007 [1]. CUDA is a C extension that allows GPU programming without passing through graphics APIs. An additional hardware was added to the GPU for this purpose. CUDA allows managing memory transfer between CPU and GPU and task scheduling for running sequential program parts on the CPU and parallel parts on the GPU. Since its apparition, many scientific and engineering applications were implemented using CUDA and high accelerations were achieved.

Even though CUDA allows GPU programming without passing through graphics API, minimum hardware knowledge is necessary for GPU programming using CUDA [2]. When an application is suitable for parallel execution, high speed-ups can be achieved over sequential execution by running it on the GPU. Usually, it is easy to obtain some speed-up for applications having data-parallelism. Nevertheless, efficient CUDA programs that achieve high speed-ups by GPU parallel

execution over sequential CPU execution require more reflection and optimization. First, it is important to efficiently handle memory operations in order to hide the long memory latency. On the other hand, it is important to have at least hundreds or thousands of concurrent threads executing in parallel in order to take advantage of the high computation throughput that could be delivered by the many GPU cores. Despite that, the acceleration that can be achieved using a GPU depends on the portion of parallel execution parts in an application. If for example $50\,\%$ of the program execution time is spent on the part that can be parallelized, $100\times$ speed-up of it will only reduce the application computation time by a factor of 1.98. On the other hand, if the computation time of the part that can be parallelized accounts for 99 % of the sequential execution time, $100\times$ speed-up of this part will reduce the application execution time by a factor of 50. Then it is crucial to maximize the portion of the program part that can be executed in parallel before doing an efficient CUDA implementation. For example, in magnetic resonance imaging (MRI) reconstruction, spiral scan data were replaced by Cartesian scan data for parallelization on the GPU [2]. Even though Cartesian data demands more computation on a CPU, it is more suitable for parallel execution. An optimized CUDA program using Cartesian scan allowed high speed-ups of more than $100\times$ to be obtained while running MRI reconstruction on the GPU by comparison to CPU. This finally reduced scanner time from the order of hours to the order of minutes.

Another important issue is to keep low complexity and high efficiency while parallelizing methods for GPU execution. An efficient GPU code will deliver high speed-ups for GPU execution by comparison to CPU execution. Nevertheless, if the resolution scheme that is used for GPU execution demands much more computation than the original sequential scheme, the GPU acceleration will be irrelevant.

Finally, GPU programmability has helped Scientists and engineers to make breakthroughs in multiple domains due to the program accelerations that were obtained. Beginning with medical imaging, it was one of the earliest applications to take advantage of GPU computing, where for example computed tomography (CT) reconstruction has reached the level where four GPUs can do the same computation that could be done using 256 CPUs [3]. Similarly, in computational fluid dynamics, very large speedups are being achieved for Navier-Stokes models and Lattice Boltzmann methods [4, 5, 6]. GPU performances are also being useful for computational finance were for example derivative pricing algorithms are accelerated up to $700\times$ [7], thus allowing more competitiveness and fast pricing results. Moreover, high speed-ups are achieved in several weather and ocean modeling applications such as numerical weather prediction [8], that enables saving in time and improvements in accuracy.

In this work, we take advantage of GPU capabilities in order to provide an extremely fast solution for computing radiation heat transfer in semi-transparent media and for computing 3D diffusion heat transfer. Work is done on the first hand in order to provide solutions methods that are highly parallelizable so they could be able to take advantage of GPU performance. Consequently, efficient GPU implementations for the solution methods are provided. On the other hand, particular effort is made in order to keep computational efficiency and low complexity of the solution methods.

Simulation of Heat Transfers in High Temperature Thermal Systems

Like in many high temperature applications, heat transfers in steel reheating furnaces are mainly due to radiative exchanges between furnace objects and to heat diffusion inside its objects and more particularly the slabs to be heated. Steel reheating furnaces are then a good example of thermal systems. Thus, they are considered in this thesis for demonstrating the fast solutions that are being developed.

In general, the difficulty of computing heat diffusion in thermal systems varies depending on boundary conditions, their variation with time and the geometry of the objects where the heat diffusion has to be computed. On the other hand, the computation of radiation exchanges is usually too computationally expensive and could be done on very large size clusters having hundreds or thousands of CPUs. Moreover, its complexity increases when coupled to conduction, convection or chemical reaction. This is because the coupling necessitates finer mesh to be applied for computing radiation heat transfer in order to provide accurate boundary conditions for other heat transfer modes.

Moreover, thermal systems control requires real time simulations, thus demanding not only high computational throughput but also more importantly fast solutions that allow the computations to be achieved in real time.

In the following some applications are considered in order to illustrate the complexity of the computation that arises primarily for computing radiation heat exchanges and then for computing heat diffusion.

Consider first the analysis of a steam explosion in a nuclear reactor obtained by the in the mixture of hot molten nuclear fuel UO_2 at 3000 K with water (figure 4). Understanding the premixing process in this reaction and knowing the water quantity in the reactor is the key for controlling the boiler process and the reactor safety. The presence of high temperatures due to the nuclear fuel implies that water temperature profile in the reactor to be highly varying. Thus, due to the high variation of the absorption coefficient of water with temperature (figure 5), radiative exchanges have to account for the highly non-gray and varying absorption coefficient in the reaction. Besides, the rapidity of the reaction implies very small computation time steps for the computation.

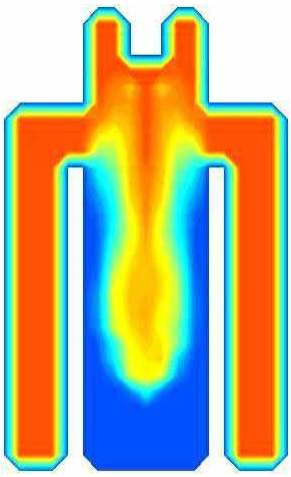


Figure 4: Steam explosion in UO_2.

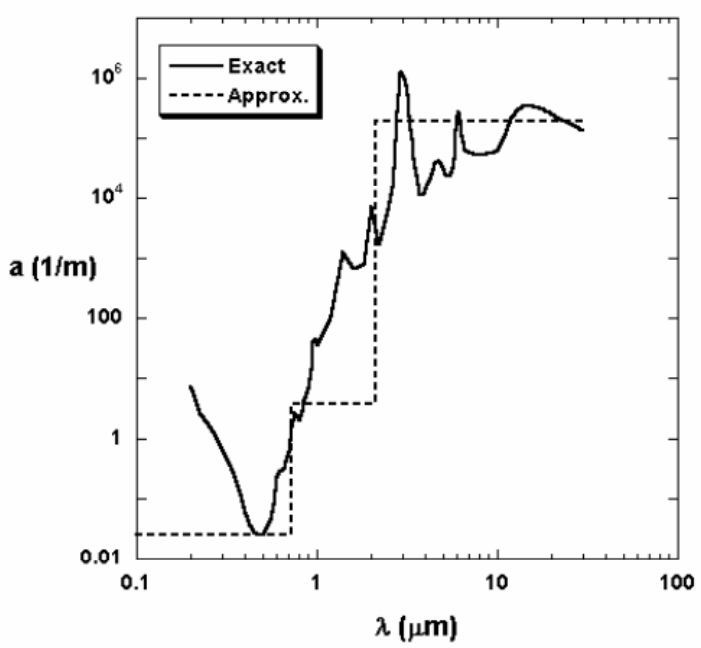


Figure 5: The 3-band approximation of water absorption coefficient used in the premixing calculation.

Besides its importance in heat transfer, radiation is the basis for illumination, which is very challenging for the video industry. Every advance in illumination techniques provides more options in games and imaging as well as it provides better and more realistic effects and images. In video games, illumination has to be computed in real-time with every change of positions and light sources. Radiation exchanges have then to be computed many times in one second in order to provide real illumination effects in the video image.

In addition to mathematical complexity, recall to the difficulty of computing radiative exchanges in inverse problems as for determining radiative properties for surfaces or volumes. Radiative properties depend on temperature, surface roughness and degree of polish and it depends on mixture proportions in gases. In inverse problems, the computation of radiative exchanges has to be repeated with different radiative emissivities and absorptivities in order to reach the input temperature profiles, which indicates that accurate radiative properties have been found. Actually, in addition to that many radiative properties have to be accounted for in one single application, thus too many repetitions of the computation are required.

Finally, consider the application case of this thesis that is steel reheating furnaces. The inline control for an industrial furnace consists of hundreds of thousands of lines of code. Nevertheless, the only too time consuming part in such programs is the computation of the heat transfer in the furnace. Due to high temperatures, the heat transfers inside a steel reheating furnace are dominated by radiation. Convection effects could be neglected. The computation of heat transfer in a steel reheating furnace consists then of computing radiation exchanges between the objects in the furnace and coupling radiation to heat diffusion inside the furnace objects, the most important being the heat diffusion inside the slabs. Computing the 3D heat transfers in the whole furnace and knowing the 3D temperature profiles of the slabs at each position of the furnace allows higher level

of furnace control. The result of that will be better heat quality and better energy efficiency. In contrast, the systems that are available to this date allow only 1D or 2D slab temperature profiles to be computed, while radiation exchanges are usually computed in advance with very large mesh size. Thus, the values of radiative exchange factors lack in precision and the mesh sizes are very large in order to provide precise boundary conditions for computing heat diffusion in the slabs.

In this work, we provide a fast solution for computing radiation heat transfer in 3D semi-transparent media based on the Multiple Absorption Coefficient Zonal Method (MACZM). In addition, a fast solution of heat diffusion is given based on an optimized GPU parallelization of the finite difference Locally One Dimensional (LOD) split method. As a result, the whole radiation exchanges in a steel reheating furnace will be able to be solved in about a second with a high precision mesh size. This is fast enough in order to allow dynamic simulation of the radiation exchanges for the inline control on the first hand and on the second hand, it provides precise boundary conditions for computing the heat diffusion in the slabs or the other furnace components. Besides, the given solution for the heat diffusion allows a high precision and very fast computation of the 3D heat diffusion in the slabs. As for example, given a time step of one second it allows to completely compute the 3D heat diffusion in a slab for the five hours time that it stays in the furnace, in less than a minute.

Manuscript Overview

Chapters 1 through 3 are dedicated for the computation of radiative exchange factors. Chapter 1 presents a numerical and experimental validation of the multiple absorption coefficient zonal method (MACZM) for computing direct radiative exchange factors in 3D inhomogeneous non-gray media. It first represents an overview of MACZM as well as the definition of Mean Beam Lengths (MBLs) that allow the computation to be done using artificial neural networks (ANNs). Then it describes the flux planes method for computing direct radiative exchange factors. Flux planes method is then used for validating MACZM for the computation of direct exchange factors between volume and surface elements. At the end of this chapter, MACZM accuracy is validated experimentally and by comparison to the flux planes method over a steel reheating test furnace. Finally, the advantage of MACZM in computation time is highlighted.

Chapter 2 covers efficient CPU and GPU implementations of the multiple absorption coefficient zonal method. In the first part of chapter 2, the algorithm corresponding to MACZM is described. Grid generation and connectivity control relative to space discretization and the method computations are discussed. Then, three efficient line discretization algorithms with linear complexity property are adapted for MACZM algorithm and compared and the best of them is kept for the implementation. In the second part of chapter 2, an efficient GPU parallel implementation of the MACZM using CUDA is represented. In this issue, GPU memory handling and efficient implementation of the different parts of the MACZM algorithm are presented. A very brief introduction to CUDA is given in that part. The reader who is totally unfamiliar with CUDA should refer first to Appendix A for better understanding of how CUDA works or skip this part directly to the application part. The last part of the chapter covers a multi-grid approach for MACZM in order to avoid unnecessary computations. The steel reheating furnace application is considered in order to describe the multi-grid approach.

Chapter 3 covers the computation of total radiative exchange factors. Even though this chapter completes the previous two chapters, there is no continuity between this chapter and the previous ones. In chapter 3, the plating algorithm for computing total exchange factors from direct exchange factors is first reviewed. A Non-Recursive Plating Algorithm (NRPA) is then formulated by identification to the plating algorithm in order to avoid recursion in computations. The NRPA is then written in matrix form that consists mainly of a matrix multiplication operation. Reduction in computational complexity for the NRPA is then achieved based on low complexity matrix multiplication algorithms. Finally, CPU and GPU implementations of the NRPA are then given based on linear algebra libraries. Besides, Multi-core CPU parallelization and GPU parallelization of the plating algorithm are presented at the end of the chapter. Sequential and parallel PA and NRPA computation time comparison is then given.

Chapter 4 is devoted for the 3D solution of the heat diffusion in slabs. First, it presents a fast review of finite difference approximation and discretization. Explicit, implicit and Crank-Nicolson schemes are reviewed. Then a review of the finite difference split methods that have less computation complexity is given. Since the split methods require the solution of tridiagonal system of linear equations, efficient sequential and parallel tri-diagonal system solvers are reviewed. A highly parallel parallelization scheme for GPU execution of the Locally One Dimensional (LOD) split method is then presented. In addition, stencil parallelization scheme is reviewed. Finally, an optimal hybrid parallelization scheme and solver strategy for running LOD method on the GPU that delivers high speed-ups is presented. The chapter ends with a simple analytical verification of the split methods that were reviewed complemented by an accuracy analysis on the influence of time and space discretization steps.

Finally, chapter 5 demonstrates an efficient thermal solution of a steel reheating furnace based on the methods and algorithms presented in the previous chapters. In this chapter, an efficient 3D numerical model for solving radiation and conduction in a steel heat furnace is first presented. Then an approach for 3D thermal analysis of the furnace is discussed. In the analysis, an approximate analytical solution is given for taking into account the cooling effect of furnace rails. The 3D temperature profiles of the slabs is then given at any time of the heating process, which makes it possible to determine the low temperature zones on the slabs, called black points. Finally, an appropriate configuration of the rails position in the furnace that results in better temperature uniformity in the slabs is determined.

REFERENCES

1. NVIDIA. CUDA Technology; http://www.nvidia.com/CUDA, 2007.
2. D. B. Kirk, W. W. Hwu, Programming Massively Parallel Processors a Hands on Approach, Elsevier inc., 2010
3. H. Scherl, B. Keck, M. Kowarschik, J. Hornegger, Fast GPU-Based CT Reconstruction using the Common Unified Device Architecture (CUDA), NSS 07 IEEE, Nuclear Science Symposium, 2007

4. J. M. Cohen, M. J. Molemaker, A Fast Double Precision CFD Code using CUDA, In 21st International Conference on Parallel Computational Fluid Dynamics, 2009.
5. Y. Zhao, Lattice Boltzmann based solver on the GPU, International Journal of Computer Graphics, vol. 24 (5), pp. 323-333, 2009
6. G. Stanvech, D. Juba, W. Dorland, and A. Varshney, Using Graphics Processors for High-Performance Computation and Visualization of Plasma Turbulence, Journal of Computing in Science and Engineering, vol. 11 (2), pp. 52-59, 2009.
7. OnEye High Performance Computing, Using GPU to Compute Options and Derivatives, OnEye Pty Ltd, Sydney, Australia, 2008.
8. J. Michalakes, GPU Acceleration of Numerical Weather Prediction, International Symposium on Parallel and Distributed Processing, IEEE, 2008

CHAPTER

Direct Exchange Factors Computation Method

1

CHAPTER CONTENTS

1.1 Introduction ..... *Erreur ! Signet non défini.*

1.2 Multiple Absorption Coefficient Zonal Method (MACZM) ..... *Erreur ! Signet non défini.*

1.3 Modray Tool Description ..... *Erreur ! Signet non défini.*

1.4 Elementary Operations Validation ..... *Erreur ! Signet non défini.*

1.5 Test Case Description ..... *Erreur ! Signet non défini.*

1.6 Calculation of Radiative Exchange Factors ..... *Erreur ! Signet non défini.*

1.7 Experimental Validation ..... *Erreur ! Signet non défini.*

1.8 Conclusions ..... *Erreur ! Signet non défini.*

References ..... *Erreur ! Signet non défini.*

EXPERIMENTAL VALIDATION OF THE MULTIPLE ABSORPTION COEFFICIENT ZONAL METHOD (MACZM) IN A DYNAMIC MODELING OF A STEEL REHEATING FURNACE

Boutros Ghannam[11], Maroun Nemer[1], Khalil El Khoury[2], and Walter Yuen[3]

[1] Mines ParisTech, CEP – *Center for Energy and Process Studies*

60, boulevard Saint-Michel – F – 75272 Paris Cedex 06

[2]*Lebanese University, Roumieh, Lebanon*

[3]*University of California Santa Barbara, Santa Barbara, CA 93106*

United States

Abstract -

In this work, the multiple absorption coefficient zonal method (MACZM) is being implemented and validated numerically. The method is demonstrated to be highly suitable for the analysis of radiative heat transfer in multi-dimensional inhomogeneous non-grey media. A uniform rectangular fine grid is considered and small CPU time is achieved. This makes the method of great interest for transient applications. The validity of the method is demonstrated in two steps. First, cases with simple geometry are considered and results are compared to results generated by direct numerical integration. Results are also generated by MODRAY, which is a source project based on an original method called the flux-planes approximation, and are shown to be equally accurate. Second, the case of a steel reheating furnace is considered. In a previous work, the furnace heat balance and temperature profiles were simulated using a finite difference computation approach and radiative exchange factors generated by MODRAY. Experiments were performed and results generated by the model were found to be in good agreement with experimental data. The radiative exchange factors are now recalculated with MACZM. They are shown to be very close to those generated by MODRAY. The comparison of the two methods clearly shows that MACZM is much faster for the calculation of the volume-volume radiative exchange factors on a uniform rectangular grid.

CHAPTER

Implementation and Parallelization of MACZM 2

CHAPTER CONTENTS

2.1 Introduction ***.....*** *Erreur ! Signet non défini.*

2.2 Multiple Absorption Coefficient Zonal Method (Maczm) and Algorithm *Erreur ! Signet non défini.*

2.3 Grid Generation And Representation of the Objects in The Discrete Space... *Erreur ! Signet non défini.*

2.4 Mean Absorption Coefficient Computation by Ray Tracing..... *Erreur ! Signet non défini.*

2.5 Artificial Neural Network and GEF Superposition..... *Erreur ! Signet non défini.*

2.6 Gpu Cuda Programming Model Overview ..... *Erreur ! Signet non défini.*

2.7 Parallel Implementation of MACZM in CUDA ..... *Erreur ! Signet non défini.*

2.8 Application: Simulation of a Steel Reheating Furnace..... *Erreur ! Signet non défini.*

2.9 Conclusions ***.....*** *Erreur ! Signet non défini.*

References ..... *Erreur ! Signet non défini.*

AN EFFICIENT CPU - GPU IMPLEMENTATION OF THE MULTIPLE ABSORPTION COEFFICIENT ZONAL METHOD (MACZM)

Boutros Ghannam[12], Maroun Nemer[1], Khalil El Khoury[1], and Walter Yuen[2]
[1]MINES ParisTech, CEP – *Center for Energy and Processes*
60, boulevard Saint-Michel – F – 75272 Paris Cedex 06
[2]*University of California Santa Barbara, Santa Barbara, CA 93106*
United States

Abstract -

The multiple absorption coefficient zonal method (MACZM) is an efficient radiative heat transfer modeling method in non-isothermal inhomogeneous media. The method is of high interest for dynamic applications because of its ability to asses semi-transparent radiative heat transfer in very short computation time. In this work, an efficient algorithm for MACZM is implemented. A connectivity control study is presented for taking into account the connectivity considerations required by the method. An identified ray traversal algorithm corresponding to part of the MACZM implementation is then selected among three different approaches presented in the paper, based on the famous ray traversal algorithms the 6-tripod line algorithm and the 6-parametric line algorithm. On another hand, the MACZM is highly parallel and is implemented in CUDA, a parallel computing architecture that enables an easy use of the powerful graphics processing unit (GPU). An efficient implementation is discussed consisting of an optimal solution for exploiting the method parallelism (threading) and the use of the memory resources available on the GPU. Speed-ups going from 300 to 600 times are achieved, using a NVIDIA Tesla C 1060 GPU and an Intel Xeon CPU E5507 at 2.27 GHz. Radiative heat transfer is then simulated in a steel reheating furnace using the optimized GPU implementation. The computation time is further reduced using a multi-grid approach.

CHAPTER

Total Exchange Factors Computation by NRPA

3

CHAPTER CONTENTS

3.1 Introduction ..... *Erreur ! Signet non défini.*

3.2 Plating Algorithm Mathematical Formulation ..... *Erreur ! Signet non défini.*

3.3 Formulation of the Non-Recursive Plating Algorithm (NRPA) ..... *Erreur ! Signet non défini.*

3.4 Matrix Form of the NRPA ..... *Erreur ! Signet non défini.*

3.5 Reducing The Computational Complexity of the NRPA Via Matrix Multiplication .. *Erreur ! Signet non défini.*

3.6 Testing The Error of Non-Recursive Plating Algorithm ..... *Erreur ! Signet non défini.*

3.7 Implementation And Computation Time Comparison of the PA And The NRPA ..... *Erreur ! Signet non défini.*

3.8 Conclusions ..... *Erreur ! Signet non défini.*

References ..... Erreur ! Signet non défini.

THE NON-RECURSIVE PLATING ALGORITHM (NRPA) FOR COMPUTING TOTAL RADIATIVE EXCHANGE FACTORS IN ENCLOSURES

Boutros Ghannam[13], Maroun Nemer[1], Khalil El Khoury[1]

[1]MINES ParisTech, CEP – *Center for Energy and Processes*

60, boulevard Saint-Michel – F – 75272 Paris Cedex 06

Abstract

Many numerical methods for computing radiation exchange in enclosures are based on the computation of direct exchange areas (DEAs) and total exchange areas (TEAs). Excessively long computation times can be associated to TEAs computation. Among the most performing methods, the plating algorithm (PA) computes TEAs from DEAs based on a set of simple recursive equations. An efficient CPU and GPU parallelization of the PA are represented. Nevertheless, PA computation complexity is $O(N^3)$. *A novel formulation, the non-recursive plating algorithm (NRPA) is introduced. It allows the computation of TEAs with one single non-recursive step. Its equations are formulated by identification to the PA equations giving TEAs from DEAs, requiring one simple assumption. The NRPA is then written in matrix form as mainly a square matrix multiplication operation. Based on advancement in matrix multiplication computation, the NRPA complexity is proven to be* $O(N^{2.38})$ *for the number of multiplications. CPU and GPU NRPA are implemented based on the optimized linear algebra library BLAS for CPU and cuBLAS for GPU CUDA programs. NRPA is found to highly outperform PA in both CPU and GPU computation times. Finally, a test enclosure is considered in order to validate the accuracy of the NRPA by comparison to the PA.*

CHAPTER

3D Heat Diffusion computation

4

CHAPTER CONTENTS

4.1 Introduction ..... *Erreur ! Signet non défini.*

4.2 Discrete Approximation of The 3d Heat Diffusion Equation by Finite Differences ... *Erreur ! Signet non défini.*

4.3 Finite Differences Split Methods Applied to the 3d Heat Diffusion PDE .. *Erreur ! Signet non défini.*

4.4 Highly Efficient Tridiagonal Linear System Solvers ..... *Erreur ! Signet non défini.*

4.5 Gpu Cuda Programming Model Overview ..... *Erreur ! Signet non défini.*

4.6 Parallelization of the Split Methods And Implementation In CUDA ..... *Erreur ! Signet non défini.*

4.7 Highly Efficient GPU Implementation of the LOD Method Using Optimal Solvers and Parallelization Schemes ..... *Erreur ! Signet non défini.*

4.8 Analytical Validation of the Split Methods ***111***

4.9 Analysis by Example of the Accuracy of the Split Methods in Order to Time and Space Mesh Size ***113***

4.10 Conclusions ..... *Erreur ! Signet non défini.*

References ..... *Erreur ! Signet non défini.*

A FAST SOLUTION OF THE 3D HEAT DIFFUSION EQUATION BY FINITE DIFFERENCE SPLIT METHODS AND AN EFFICIENT GPU PARALLELIZATION

Boutros Ghannam[14], Maroun Nemer[1]

[1]Mines ParisTech, CEP – *Center for Energy and Processes*

60, boulevard Saint-Michel – F – 75272 Paris Cedex 06

Abstract

In this paper an optimal and efficient numerical solution of the 3D heat diffusion Equation on GPUs is presented. The solution is the result of the optimized combination between finite difference discretization scheme, parallelization schemes in CUDA, and appropriate tridiagonal matrix solvers. Finite difference split methods are used for their low computational complexity and parallelization property. At the first hand, a new parallelization scheme is applied for the solution of the Locally One Dimensional (LOD) method. The scheme is shown to give high accelerations going up to 60 times by comparison to CPU time, beginning from small grid size. On the other hand, Thomas algorithm for the solution of linear tridiagonal systems is used for the first time, together with a stencil parallelization. It is shown to deliver higher performance by comparison to the first scheme with large grid size, with up 250 times acceleration by comparison to CPU time. Finally, a combined alternating parallel scheme solution is discussed, allowing an optimal performance to be achieved even for small grid size.

4.8 ANALYTICAL VALIDATION OF THE SPLIT METHODS

4.8.1 Setting the problem

A cube of dimensions $1m \times 1m \times 1m$ is considered for the validation (Figure 14). The cube is at initial temperature $T_\infty = 300\ K$ and exchanges heat by convection to a heat source of temperature $T_\infty = 1300\ K$. A constant convection heat exchange coefficient $h = 110\ W.m^{-2}$ is considered on all sides of the cube. Because of an analytical solution, the physical properties of the cube are considered to be constant during the whole heating time, giving a conductivity $k = 55\ W.m^{-1}K^{-1}$, a heat capacity $c_p = 520\ KJ.Kg^{-1}$, and density $\rho = 7800\ Kg.m^{-3}$.

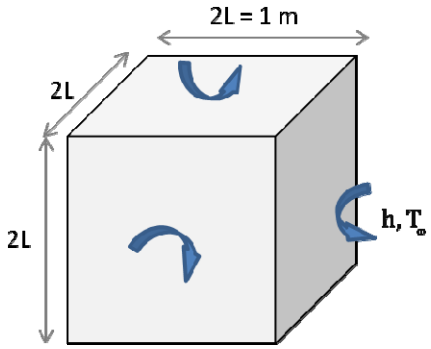


Figure 14: Cube submitted to convection heat flux

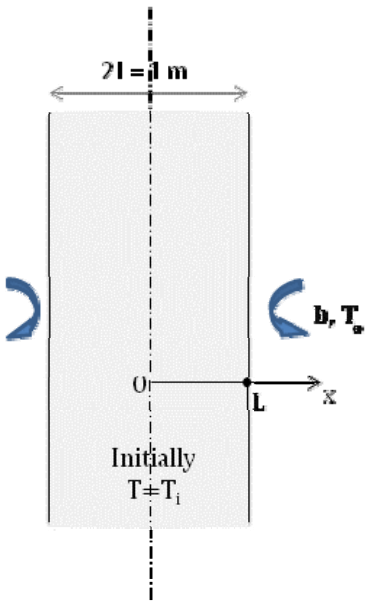
4.8.2 Analytical solution

The analytical solution for the considered three-dimensional case is the product of three one-dimensional analytical solutions of the transient heat conduction in a plain wall (Figure 15).

The governing PDE for the one-dimensional heat diffusion is as follows:

$$\frac{\partial T}{\partial t} = \alpha \frac{\partial^2 T}{\partial x^2} \tag{44}$$

Where $\alpha = k/\rho c_p$ is the diffusivity of the material.



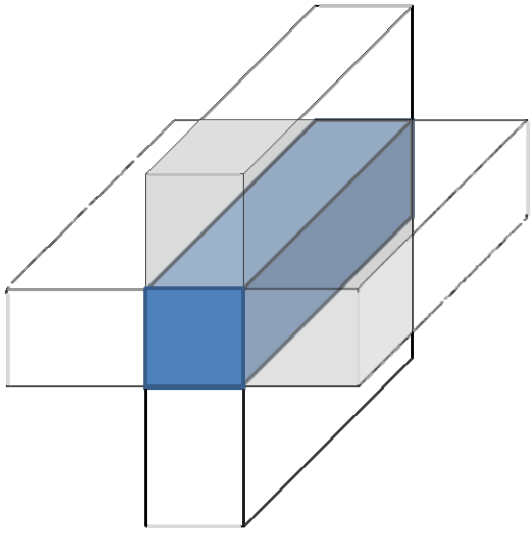


Figure 15: Plane wall

Figure 16: Product of two plane wall solutions

The convective boundary condition for the plain wall of thickness $2L$ is:

$$-k\frac{\partial T(L,t)}{\partial x} = h(T(L,t) - T_\infty) \tag{45}$$

Writing Equations (44) and (45) in dimensionless form gives the following:

$$\frac{\partial^2\theta}{\partial X^2} = \frac{L^2}{\alpha}\frac{\partial\theta}{\partial\tau} \tag{46}$$

And

$$\frac{\partial\theta(1,\tau)}{\partial X} = -Bi\theta(1,\tau) \tag{47}$$

Where

$\theta(X,\tau) = \dfrac{T(x,t) - T_i}{T_\infty - T_i}$ Dimensionless temperature

$X = \dfrac{x}{L}$ Dimensionless distance from the center

$Bi = \dfrac{hL}{k}$ Dimensionless heat transfer (Biot number)

$\tau = \dfrac{\alpha t}{L^2}$ Dimensionless time (Fourier number)

The analytical solution for the temperature profile in the plain wall is given as a function of the dimensionless variables as:

$$\theta(X,\tau) = \sum_{n=1}^{\infty}\frac{4\sin\lambda_n}{2\lambda_n\sin(\lambda_n)}e^{-\lambda_n^2\tau}\cos(\lambda_n X/L) \tag{48}$$

Where λ_n values are the solution of $\lambda_n \tan\lambda_n = Bi$.

This solution can be approximated by its first term for $\tau > 0.2$ with an error that is less than 2%. Thus if we consider the temperature at the center plane of the plain wall, it is given by:

$$\theta(0,\tau) = \frac{4\sin\lambda_n}{2\lambda_n\sin(\lambda_n)}e^{-\lambda_n^2\tau} \tag{49}$$

For $\tau > 0.2$.

Finally, the analytical solution for the temperature at the center point of the cube is the product of the solutions of three identical solutions of the temperature at the center plane of a plain wall (Figure 16). The solution is then given by:

$$\theta(0,\tau)_{cube} = \theta(0,\tau)_{wall} \times \theta(0,\tau)_{wall} \times \theta(0,\tau)_{wall} \tag{50}$$

Where for the cube $Bi = 1.0$, and then $\lambda_1 = 0.8603$ and the solution is:

$$\theta(0,\tau)_{cube} = \frac{T(center,t) - T_i}{T_\infty - T_i} = \left(1.1191e^{-0.8603^2\tau}\right)^3 \quad (51)$$

The analytical solution is given in Figure 17 for $t > 7000s$, in order to guarantee that $\tau > 0.2$.

Comparison to numerical solution

The temperature is computed by all the split methods using grid size $32 \times 32 \times 32$ and a time step of $10s$. The resulting solution is presented in Figure 17 for $t > 7000s$ in order to compare it to the analytical solution. As it could be observed on Figure 17, all numerical solutions are very close to the analytical solution. The error of the numerical solutions is always less than 1.5% by comparison to the analytical solution, which demonstrates the validity of the solution that were given by the split methods.

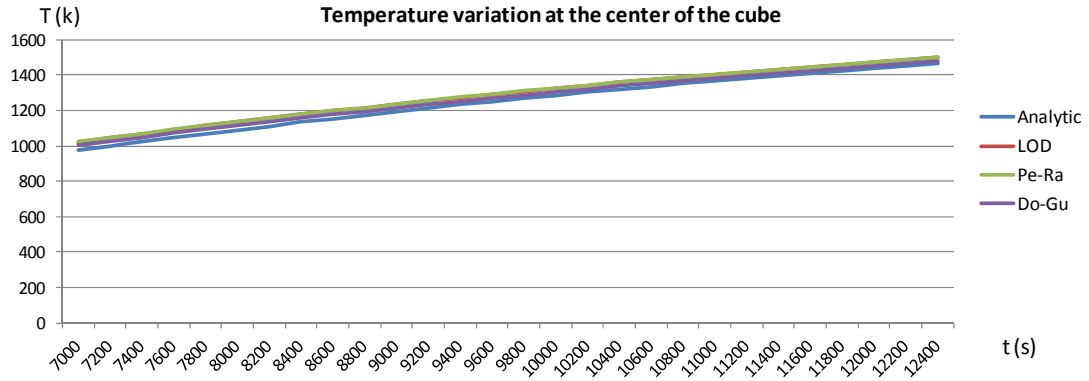


Figure 17: Comparison of numerical solutions to the analytic solution

4.9 ANALYSIS BY EXAMPLE OF THE ACCURACY OF THE SPLIT METHODS AS A FUNCTION OF TIME AND SPACE MESH SIZE

4.9.1 Numerical simulation

Consider a steel slab of size $0.8m \times 1.6m \times 0.25m$. The slabs lays on two supports as presented in Figure 18. The slab is heated by the surface flux profile of Figure 19. The heat flux is supposed to be null at the contact with the supports. The slab temperature is computed separately by the different split methods over a period of 15000s. The physical properties in the slab vary with

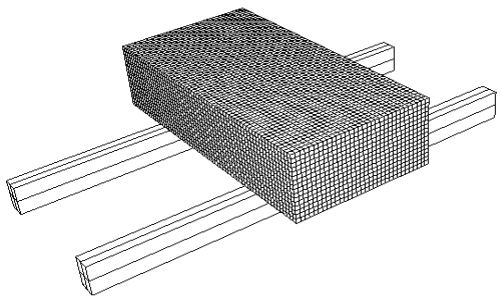


Figure 18: Slab laying on two supports

the temperature during the simulation. A very fine meshing is first considered consisting of $128 \times 128 \times 128$ space mesh and a time step of 0.1 second. The resulting variation of the temperature at the center of the slab and at the quarter of the height over the support that were obtained by the different split methods are presented in Figures 20.a and 20.b. The slab is then simulated by each method using different space and time steps. The results of the simulations at the center of the slab are presented in Figure 21 (21.a-21.e).

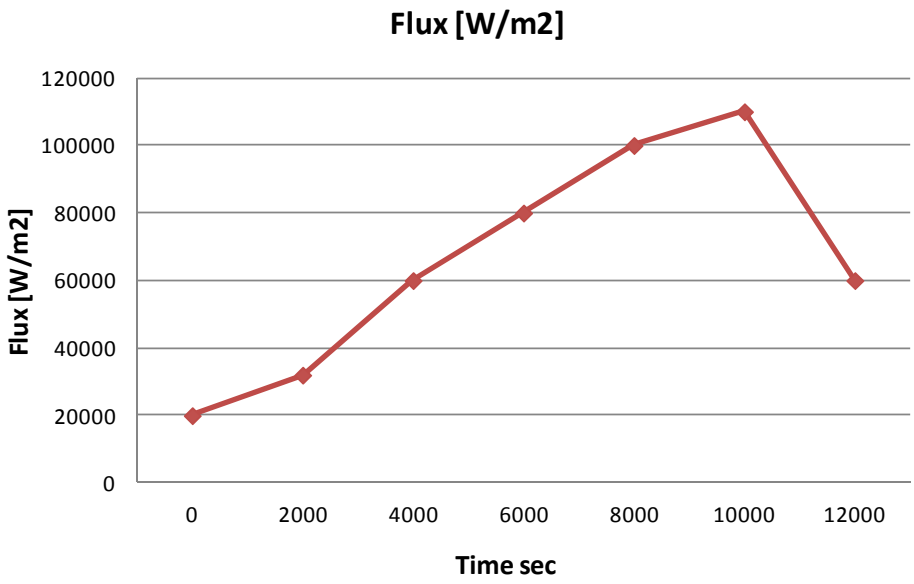


Figure 19: Heating flux as a function of time.

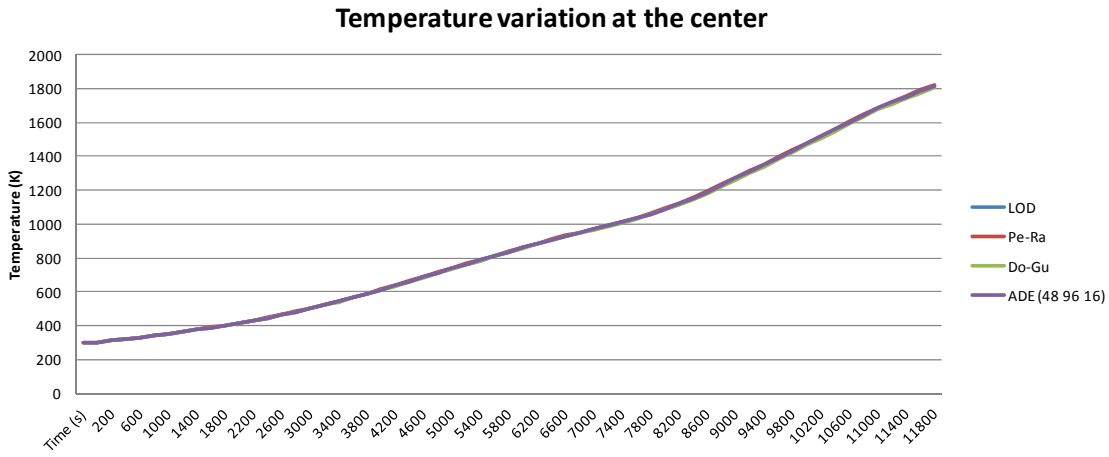


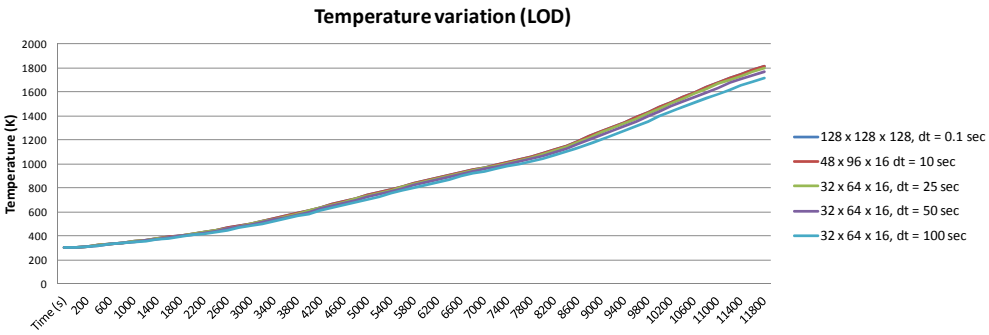
Figure 20: Temperature increase at the center of the slab as a function of time.

4.9.2 Comparison and analysis

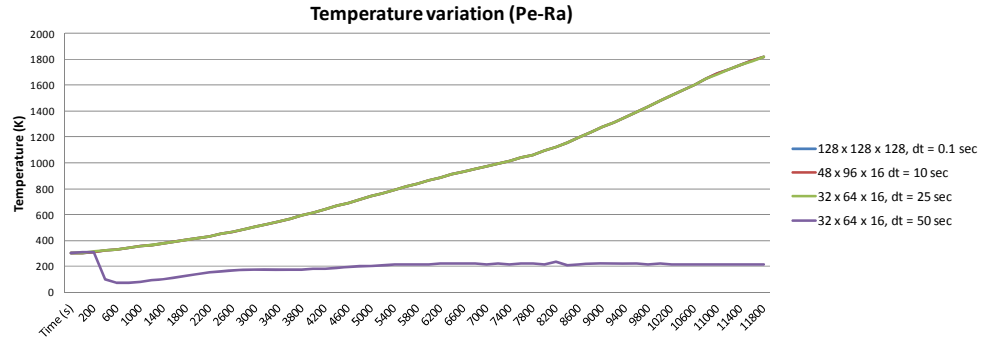
In Figure 20, for a fine meshing, all methods give the same temperature profile. This was expected and it validates the consistency and convergence property of the different methods. On the other hand, Figure 21 demonstrates the stability and the accuracy of each method. As it could be observed, the LOD method (Figure 21.a) and the Douglas-Gunn ADI method (Figure 21.b) give the

most accurate results with lower space and time mesh. This is because they are unconditionally stable and second order accurate in time and space. As for the ADE method (Figure 21.c), it is less precise when lower space and time steps are considered, even though it is still stable with large time steps. On the other hand, the Peaceman-Rachford ADI method (Figure 21.d) is first order accurate in time, thus its error becomes larger when larger time step is considered and it becomes unstable when for very large time steps.

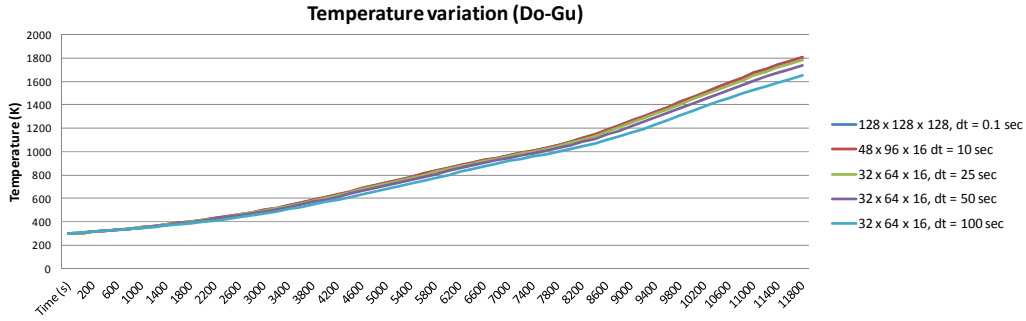
(a)



(b)



(c)



(d)

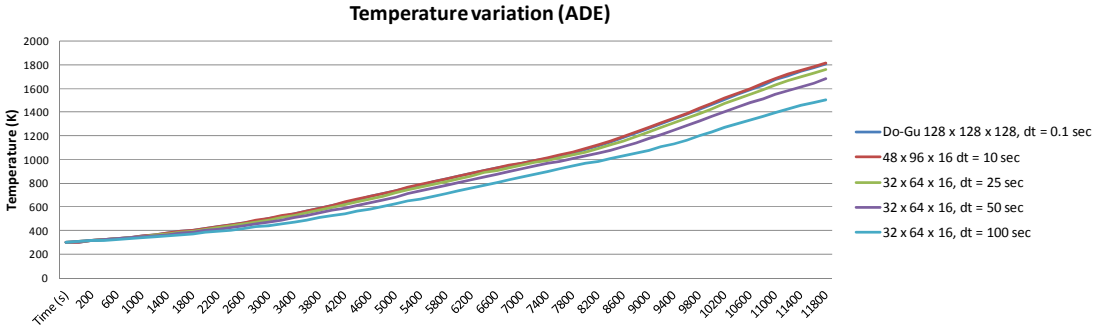


Figure 21: Comparison of the split methods temperature increase at the center of the slab for different space and time mesh.

CHAPTER

3D Ultra-Fast Simulation of Steel Reheating Furnace, Opening for Inline Control

5

CHAPTER CONTENTS

5.1 Introduction 117

5.2 3d Numerical Computation Model for a Steel Reheating Furnace And Performance Evaluation 120

5.3 Application: Thermal Analysis of a Steel Reheating Furnace 125

5.4 Optimization of Rails Positions and Heating Temperature Profiles 127

5.5 Conclusions and Future Work 131

NOMENCLATURE

A	Surface area	Q	Heat flux
k	Thermal conductivity	T	Temperature
F	Transfer factor		
I	Zone number	***Greek symbols***	
N	Number of furnace patitions	σ	Stefan-Boltzmann constant

5.1 INTRODUCTION

Applications where the radiative heat exchange is dominant are very challenging. Steel reheating furnaces are a good example of application where the radiative heat exchange is dominant, because of the high operating temperatures. A steel reheating furnace is filled with non-homogeneous media from combustion gases. Steel slabs circulate in the furnace in order to be reheated to temperatures around 1200℃. After the slabs are heated, they are submitted to a post treatment process as rolling for example. In order to achieve a good quality of the post treatment process, the temperature profile of the slabs leaving the furnace must correspond to a specific set point. In the case of rolling for example, a uniform temperature profile is needed. In addition, the heating profile of the slabs is

also object to a set point in order guarantee better energy efficiency on the first hand and achieve the desired temperature profile in the slabs at the end of the furnace on the second hand.

Heating temperature profile set point and final temperature profile of the slabs can be achieved using inline controllers. Because of the high operating temperatures and the slabs movement in the furnace, it is impossible to continuously measure the slabs temperature in the furnace. Thus, it is necessary to numerically compute the temperature profiles in the slabs and the rest of the furnace. Moreover, with numerical computation it is possible to make computations for the next minutes or hours of heating before reaching them, which is crucial in order to build efficient inline control strategies.

In contrast, the numerical solvers that were developed earlier lack in speed and precision. They usually are too slow in order to compute inline temperature profiles in the furnace. In consequence, simplifications are usually made, considering of computing the heat diffusion in the slabs in 1D or 2D and setting a low-resolution mesh for computing radiative heat exchanges. On the other hand, the temperature profiles for the forthcoming operating time of the furnace are only predicted in order to be used for the inline control, instead of being precisely computed.

In this chapter, first a detailed numerical solver for the heat exchanges in the furnace is presented based on the methods developed in the previous chapters. The multi-grid approach is used for computing the direct exchange factors. In addition, computation of the insignificant direct exchange factors is eliminated. A zone approach is then introduced for applying the NRPA in order to make the computation of the total exchange factors as fast as the computation of the direct exchange factors. The radiation is then coupled to heat diffusion, which is computed using the efficient LOD scheme presented in chapter 4. As a result, a high precision 3D numerical solver is set and very fast computation time is achieved, that allows inline dynamic computation and computation of the forthcoming functioning time.

A thermal model for solving the heat exchanges in the furnace is then presented. Using this model, an analysis of black points is given based on the configuration of the positions of the rails. Black points are the zones of the slab where the temperature is lower due to the cooling by the contact with the supporting rails. On the other hand, a wall temperature profile is determined that allows a desired delayed heating temperature profile to be achieved in the slabs.

5.2 3D NUMERICAL COMPUTATION MODEL FOR A STEEL REHEATING FURNACE AND PERFORMANCE EVALUATION

Consider the steel reheating furnace of chapter 2.8. The internal dimensions of the furnace are $36.275\,m \times 6.8\,m \times 3.8\,m$. Figure 1 shows the elements inside the furnace. The furnace can hold on average 45 slabs of steel laid on four rails. The slabs of steel have different lengths but they all have a height of $0.2\,m$ and a width of $0.8\,m$ (Figure 1).

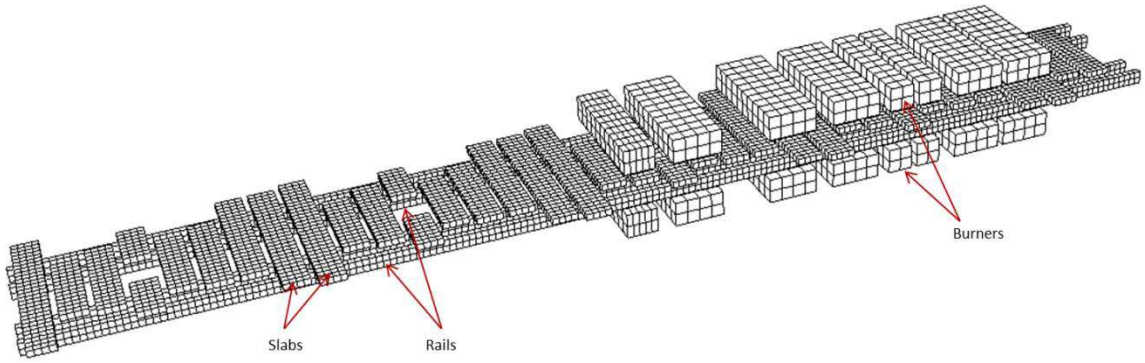


Figure 1: Inside elements of the steel reheating furnace

An average distance of 0.2 m is supposed to be separating the steel slabs over the length of the furnace. The rails are assumed to have a rectangular section of size 0.2 m × 0.4 m. The first part of the furnace is a preheating zone with no burners where the slabs receive heat flux from the hot gazes coming from the heating zone.

The heating zone is equipped by regenerative burners that are symmetrically positioned in the upper and lower zones of the furnace. All burners are identical and the combustion volumes are assumed to be rectangular with dimensions 0.8 m × 4 m × 0.8 m. An emissivity of 0.9 is assumed for the furnace walls. The slabs are assumed to have an emissivity of 0.8 and the rails an emissivity of 0.75. The volume of the furnace is filled with combustion gases that are supposed to be acting as gray-diffusive and non-scattering with an absorptivity of 0.5 m^{-1}.

5.2.1 Computation of the direct radiative exchange factors

The direct radiative exchange factors are computed by the Multiple Absorption Coefficient Zonal Mehtod using the multi-grid approach presented in Chapter 2.8.5. The multi-grid approach implies that each category of exchange factors is computed using an appropriate mesh size in order to guarantee a good accuracy and to avoid any unnecessary calculation time. A mesh size of 40 cm^3 (0.15 Mvoxels) is applied to the furnace in order to compute the direct exchange factors between the burners and the direct exchange factors between the furnace walls. Direct exchange factors between walls and steel slabs, rails and burners as well as between burners and walls are computed using a mesh size of 20 cm^3 (1.2 Mvoxels). At last, a mesh size of size 10 cm^3 (over 9 voxels) is considered in order to compute direct exchange factors between the slabs themselves and between slabs and rails. A larger mesh being less accurate for computing the former exchange factors because of the dimensions and the relative position of the slabs to the rails and the slabs between them. At this level, a CPU computation time of 1200 seconds is achieved for one single of the direct exchange factors in the furnace, while GPU execution is about 320 times faster with a computation time of 3.6 seconds, using an Intel Xeon CPU E5507 at 2.27 GHz and a NVIDIA Tesla GPU C 1060.

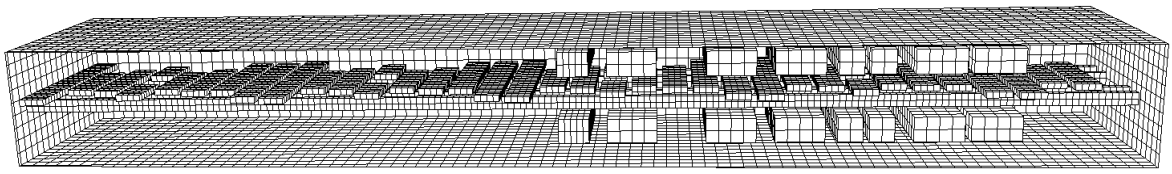


Figure 2: Final mesh for the computation of radiation exchange factors

At the end, the computed direct exchange factors are associated together in order to give for each object the highest dimension of mesh that were applied to it during the computations. Figure 2 shows the final meshing of the furnace. The final mesh size is 20 cm^3 for steel slabs and the rails while a mesh size of 40 cm^3 is given to the burners, the furnace volume and walls.

Further saving in computation time is done by eliminating the computation of insignificant direct exchange factors. Because of the low ratio of the width and height of the furnace to its length, the direct exchange radiation leaving a surface element at for example the middle of the furnace is absorbed long before it reaches the beginning or the end of the furnace (Figure 3).

The limiting distance from which the radiative exchange is negligible is determined dynamically while computing the radiative exchange factors. Beginning the computation from neighboor vertical planes and going up to distant planes, the computation is stopped when the summation of the direct exchange factors is at 99% of its value. As a result, a CPU computation time of 180 seconds and a GPU time of less than a second are achieved.

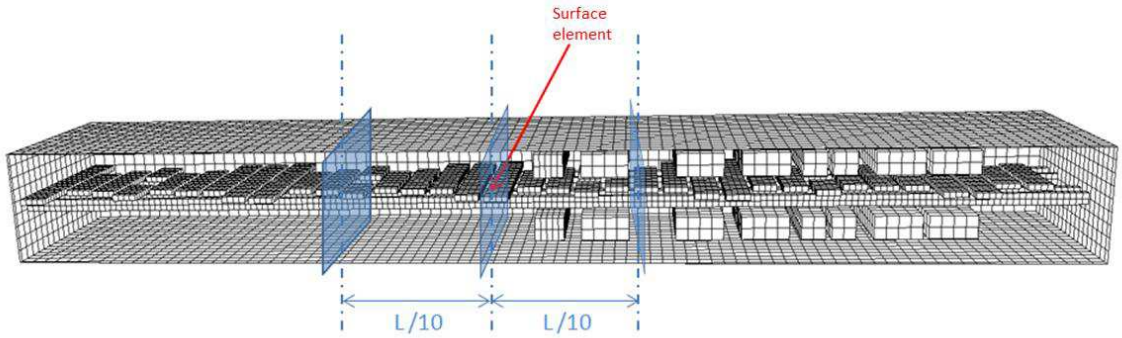


Figure 3: Limiting the computation of direct exchange factors over the length of the furnace

5.2.2 Computation of the total radiative exchange factors

Given the emissivities of the furnace surfaces and the direct exchange factors between all the surface elements, the total exchange factors are computed using the plating algorithm. The total number of surface elements in the furnace after the multi-grid approach is applied is 14660. Applying the plating algorithm directly to the furnace with this number of surfaces elements would necessitate about two hours CPU computation time, which is too slow by comparison to the computation of the direct exchange factors.

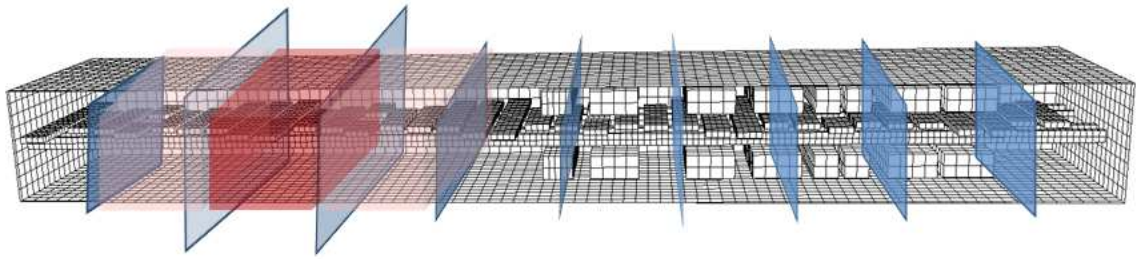


Figure 4: Zone computation of the total exchange factors

On the other hand, applying the NRPA to compute the total exchange factors requires a GPU computation time of 9 seconds at order 2 and 17 seconds at order 3. The corresponding weighted average errors by comparison to the plating algorithm being 3% and 1 % respectively for order 2 and order 3 NRPA respectively.

The NRPA computation time is still an order of magnitude higher than the computation time of direct exchange areas. Thus, it is still the limiting factor for the computation time. As a solution, the total exchange factors could be approximated by dividing the furnace to a number of computation zones. Applying the NRPA to zones with lower number of surface elements results in faster computation time since the NRPA is of order $O(n^{2.38})$. The zone computation is shown in figure 4.

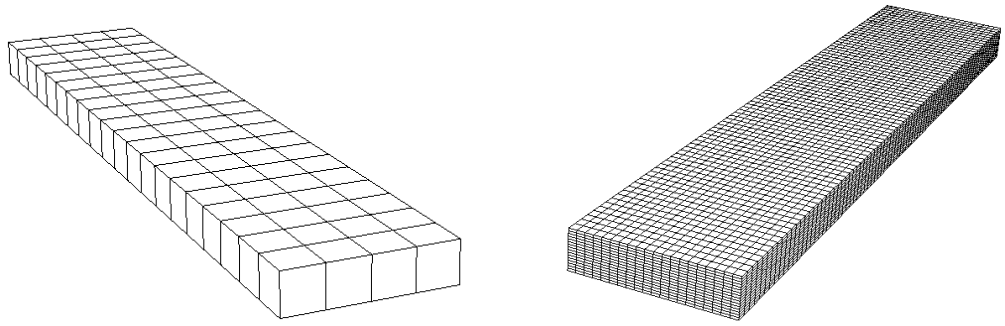
The furnace is first divided into N zones in the length direction. The total exchange areas of a zone I are then computed by taking into account the plating of surface elements of the zones $I-1$, I and $I+1$. Using N=10, a computation time of 0.75 seconds is achieved by the order 3 NRPA with an additional error varying between 1 and 2 % over the computed total exchange areas. In this case, the NRPA is applied to systems with lower number of surfaces n.

5.2.3 Computation of temperature profiles in the slabs

A slab temperature profile is given by computing the heat diffusion in the slab given the adequate boundary conditions. The heat diffusion in the slabs is computed by the LOD method (Chapter 4.3.1). A fine mesh is applied to the slabs for the computation. Neglecting convection, the boundary conditions on the slabs are flux boundary conditions due to the radiation between the slab surfaces and other surfaces and volumes in the furnace and to the heat conduction to the rails.

Even though the mesh considered for computing radiative exchange factors is fine and sufficiently precise, the corresponding mesh size still is larger than the mesh that is applied for computing heat diffusion by finite differences (Figure 5). Consequently, the computed radiative heat flux is projected on the diffusion mesh in order to give the flux boundary condition. If more than a radiative computation mesh intersect on a diffusion computation mesh area, the corresponding radiative heat flux is computed as the average of the intersecting radiative fluxes, each of them being weighted by the proportion of its corresponding surface area proportion on the diffusion mesh.

A central first order finite difference approximation of the flux boundary conditions is considered for applying the LOD method in order to guarantee that the result is still second order accurate.



(a) (b)

Figure 5: Coupling radiation and diffusion (a) Slab mesh for computing radiation exchange factors (b) Slab mesh for diffusion computation.

The computation is done using the highly efficient GPU implementation given in chapter 4.7. It achieves a very fast computation time. For example, using a grid of $32 \times 64 \times 16$ meshes, allows achieving a computation time lower than 1/3000 seconds by time step. A highly accurate 3D temperature profile in all the slabs is finally obtained.

5.2.4 Dynamic simulation and performance evaluation

The slabs are moving at a speed of $0.0025\, ms^{-1}$. Maximum precision for the simulations is obtained if the computation of the radiative exchange factors is done for the smaller possible distance step in the slab moving direction. From the previous section, the largest mesh size used for computing direct exchange factors between a surface element on a slab and any other surface element in the furnace is $20cm$. This implies that the minimum distance step that could be considered for computing radiative exchange factors is $20cm$. The radiative exchange factors should then be computed for a distance step of $20cm$ or equally a time step of 80 seconds. The computation time that is needed to achieve the computation of radiative exchange factors relative to one time step is about 1.75 seconds (NVIDIA Tesla GPU C 1060). A time step of 80 seconds is then large enough to allow real-time dynamic simulation of the furnace since it is much higher than 1.75 seconds.

The time step for radiative exchange factors computation is then 80 seconds, which implies that heat diffusion will have to be computed using a time step that is smaller than 80 seconds. Even if the radiative exchange factors are constant over a 80 second time period; the radiative heat flux on the slabs could be varying during this period, since the temperatures of the slab and the furnace are varying continuously.

A grid of $32 \times 64 \times 16$ meshes for the computation of heat diffusion in the slabs will keep a good precision and a fast computation time. From the diffusion computing section, the computation time corresponding to this mesh is $1/3000$ seconds by slab and by time step. Then computing the diffusion in all the 45 slabs of the furnace will require 0.015 seconds, defining then the minimum time step for computing heat diffusion in the real-time dynamic simulation. Obviously, much higher time steps would be considered for the heat diffusion computation, since the accuracy of the results will not be significantly affected.

In summary, real-time dynamic simulation is done using a $80s$ time step for computing radiative exchange factors and a time step for diffusion computation that is higher than $0.015s$ (NVIDIA Tesla GPU C 1060).

Current inline control is applied to the furnace should be updated each 10 seconds taking into account that a time step of 80 seconds is imposed for radiative exchange factors and a time step of 5 seconds for the heat diffusion computation. The computation time that is needed for virtual movement of the slabs of $20cm$ will then be 1.75 seconds for radiative exchange factors computation and 0.24 seconds for heat diffusion computation using an NVIDIA Tesla GPU C 1060, making about 2 seconds for computing the radiation and heat diffusion for the slabs being moved of a distance of $20cm$. As a result, a predictive computation for the slabs moving a distance of $1m$ could be done in the 10 seconds period preceding each update of the furnace inputs. This performance is directly related to the GPU performance. Considering a newer GPU, the NVIDIA GeForce GTX 690 for example, it has two GPUs with a sum of 3072 cores while the C 1060 has only 240. The GTX will easily deliver five times higher performance than the C 1060. In consequence, if a computing station with two NVIDIA GeForce GTX 690 GPUs is used, a 20 meters prediction computation for the slabs movement could be done during the 10 seconds period after which the furnace inputs have to be updated. This computing throughput would have necessitated a CPU cluster with approximately a thousand of CPU cores.

5.3 APPLICATION: THERMAL ANALYSIS OF A STEEL REHEATING FURNACE

5.3.1 Thermal analysis model

The same furnace is now considered. Many thermal analysis models could be set for a steel reheating furnace depending on input temperatures or heat fluxes. In the same way, assumptions could be taken for the temperature profiles in the walls or the gases or the combustion volumes. In this work, the wall temperature is set to be the input for the thermal analysis and the furnace is supposed to be heated due to high temperature gases filling the furnace without burners. The temperature of the walls is assumed to be uniform over vertical sections. This gives a one-dimensional temperature profile of the walls in the direction of the length of the furnace. The wall temperature profile is given as input to the thermal model at any time of the computation. The temperature of the gases in the furnace is also assumed uniform in vertical planes, thus it varies only over the length of the furnace. The rails are cooled with a large flow of water flowing from the beginning to the end of the furnace. The output water temperature is slightly higher than the inlet temperature and thus a mean temperature T_w between the inlet temperature and the output temperature of the water is assumed to be accurate for the computations.

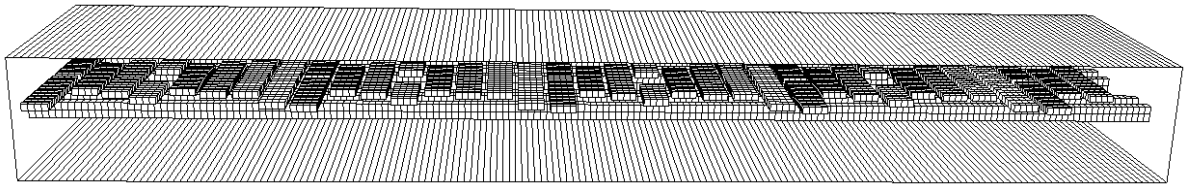


Figure 6: Furnace mesh for thermal analysis.

5.3.2 Discretization

The furnace discretization for computing radiative exchange factors is changed by comparison to the previous section in order to correspond better to the assumption of uniform temperature in the vertical planes for the walls and gases. All the radiative exchange factors are computed using the multi-grid approach and NRPA of order 3 as described in the previous section. A maximum voxel size of 20 cm^3 is considered for computing the direct radiative exchange factors and a partial grouping of the direct exchange factors is applied in order to compute the total exchange factors. The total radiative exchange factors are then grouped for the thermal simulation correspondingly to the assumption of uniform temperature in vertical planes for the walls and the gases. As shown in the figure 6 a mesh size of 20 cm is kept for the walls and the gases all over the length of the furnace. Instead, all the wall and gas meshes are grouped into one mesh for walls and one mesh for the gases in the vertical planes since no meshing is necessary because of the uniform temperature assumption. Finally, 3D surface mesh of size 20 cm^2 is kept for the slabs and the rails.

5.3.3 Radiation heat exchange in the furnace

The radiation heat flux between any couple of meshes of the furnace is given by:

$$Q_{12} = A_1 F_{12} \sigma (T_1^4 - T_2^4) \quad (1)$$

Where, A_1 is the surface area of the first mesh. F_{12} is the total exchange factor between the first mesh and the second mesh. σ is Stefan Boltzmann constant. T_1 and T_2 are the temperature of the first mesh and the second one respectively.

5.3.4 Thermal model of the walls and the rails

The walls exchange heat to the inside of the furnace by radiation. They also dissipate heat from the furnace. The heat dissipation via the walls occurs by heat conduction through the walls to the outside. On the other hand, each rail is cooled by water flowing through two circular pipes. Thus, they absorb energy from the slabs by conduction and from the other components of the furnace by radiation.

Since it is not the object of this work, a simple model is applied for approximating heat transfer in both the walls and the rails. Heat conduction through the walls is assumed to be at steady state and two-dimensional in vertical planes. It is then computed as the heat conduction through a plain wall for each vertical mesh. Nevertheless, it could be neglected since the walls are very thick and their conductivity is very low. Thus, the heat dissipation through the walls is very small compared to the energy flow inside the furnace.

A simplified geometry of the pipe is shown in figure 7. Each rail is cooled by water flowing through two circular pipes. An insulator separates the pipes from the outside material of the rails. A steel part separates the upper pipe from the slab and is in direct contact with this pipe. The heat transfer in the rails is assumed to be two-dimensional in the vertical plan and steady state. Using these assumptions and the simplified geometry of the rail, the heat transfer between from the rails by conduction and radiation is determined analytically by computing the equivalent thermal resistances through the rails.

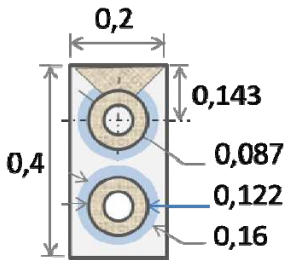


Figure 7: Rail cross section

5.3.5 Energy conservation and gas temperature

In order to ensure energy conservation in the furnace, walls energy balance is computed at each time step. Walls transfer heat to the slabs and the rails and they receive heat from the gases inside the furnace and they also dissipate heat to the outside. The energy transferred from the walls to the slabs, the rails and the outside is taken from the gases. Since the temperature of the walls is known (imposed) and the temperatures of the slabs and the rails are given from a previous time step computation, the temperature of the gases is the only unknown and its value should verify the energy equilibrium on the walls. The temperature of the gases can then be computed. Finally, knowing the gases temperature allows to compute the heat transfer from the gases to the slabs and the rails.

5.4 OPTIMIZATION OF RAILS POSITIONS AND HEATING TEMPERATURE PROFILES

5.4.1 Thermal model inputs

The ambient temperature outside the furnace is $T_a = 300\ K$. The walls have a thickness of 50 cm and they are made of brick. The water flow in each rail is 300 m^3h^{-1}. The slabs circulates through the furnace at a speed of 0.0025 ms^{-1}. The cooling water enters the rails at a temperature of $303K$ (30°C) and leaves at $312K$ (39°C). Water temperature is then assumed to be $T_w = 308\ K$. The rails insulation has a conductivity of 0.25 $Wm^{-1}K^{-1}$. The rails pipes and the steel part separating the upper pipe from the slabs are assumed to have a conductivity of 35 $Wm^{-1}K^{-1}$ while the rest of the rails has a conductivity of 25 $Wm^{-1}K^{-1}$.

5.4.2 Black points analysis and rails positioning

Here we are going to consider the temperature profile of a slab of typical dimensions 80 cm × 400 cm × 20 cm during its journey in the furnace. The temperature profile of the walls is determined as follows. The first part of the furnace is considered to be as a preheating zone. According to this the temperature of the walls at the beginning is low and equal to $800K$. After a distance of 6 meters the temperature of the walls grows fast in order to reach a maximum of $1600K$.

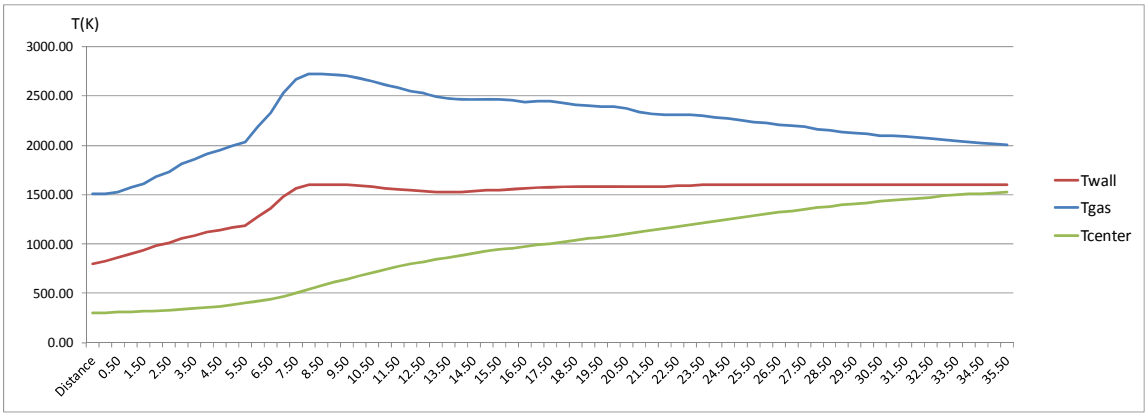


Figure 8: Temperature profiles as a function of furnace length.

After that, the temperature of the walls is determined in a manner to keep it maximal while not exceeding a temperature difference of 120 K at any vertical line in the considered slab. This allows the slab to be heated as fast as possible. It consequently maximizes the black points effect due to rail cooling. In figure 8 are shown the temperature profiles of the walls and the gases and the temperature at the center of the slab as a function of the furnace length. From the figure, it could be seen that heating profile in the slab is similar to an extended heating profile after the preheating zone. Figure 9 shows the maximum vertical temperature differences that occur in the slab as function of the position in the furnace length.

Three configurations for the positions of the rails in the furnace are now considered (Table 1 A). Wall temperatures are determined similarly and the temperature profiles in the middle of the slab at at a height of $5cm$ are shown in Figure 10. Case A is identical to the furnace shown in Figure 1. In this case the four rails are parallel, two on each side of the furnace and their position is constant over all the furnace length. From figure 10.a it is clear that the temperature of black points in this case is very low relatively to the temperatures in other zones of the slabs. This is because high cooling is applied to the slab from the rails and it is concentrated at the same points since the rails position under the slab is constant over the whole furnace. The rails positions have then to be varied over the furnace in order to avoid concentrated cooling.

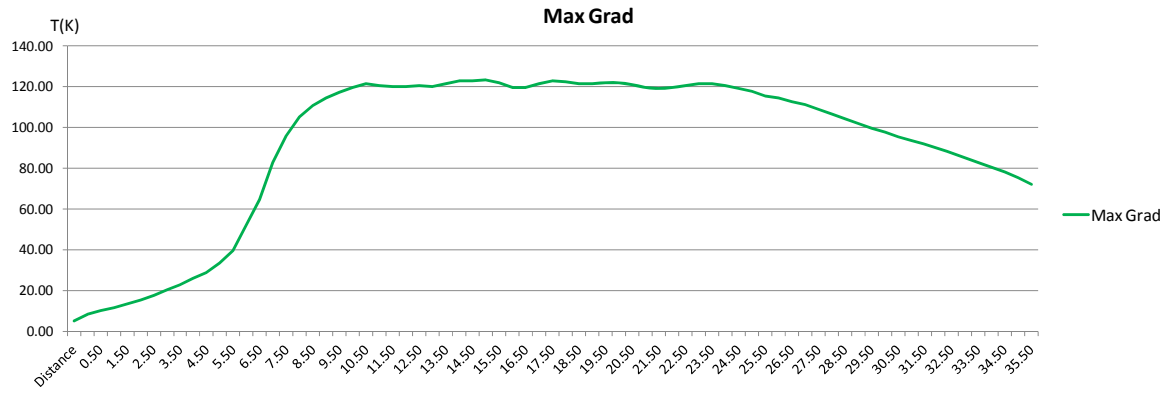
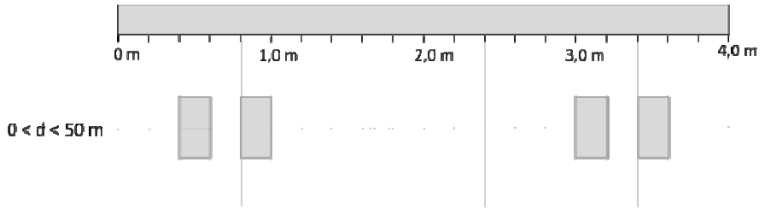
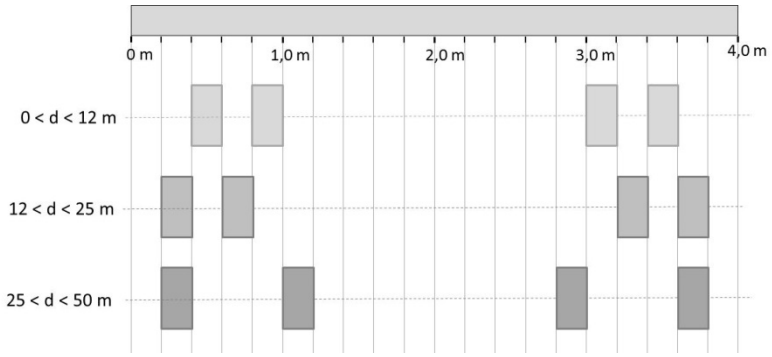
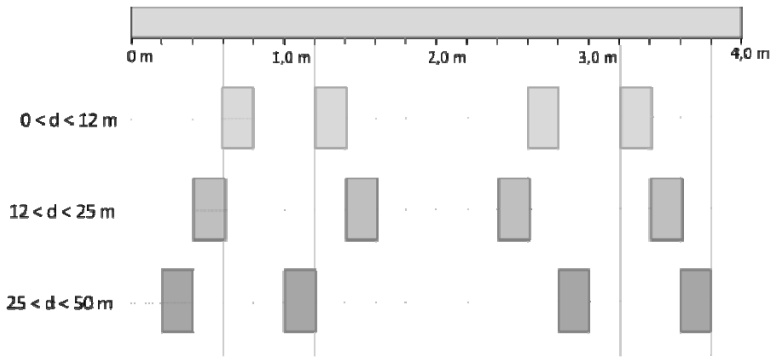


Figure 9: Maximum temperature difference in the thickness of the slab as a function of furnace length.

Table 1 - Different cases of configuration for the positions of the rails

A.	
B.	
C.	

Besides, the distance between the rails has to be smaller when the slab temperature gets higher through the furnace in order to avoid high bending. Considering the preceding two constraints, two other configurations for the rails positions are then considered (Table 1 B and C). In case B, the rails position is changed two times through the length of the furnace in order to avoid concentrated cooling of the slabs. From figure 10.b it could be seen that the temperature for the black points for case B is higher than case A and it approaches more the temperature of the other zones of the slab. Nevertheless, the temperature of the slab is still much higher at its center and near the borders where is receives additional radiation flux coming from the vertical walls of the furnace. A good strategy is then to move the rails near to the center and the borders of the slab while still changing their positions through the furnace. This is done in case C. As a result, the black point effect is lower and the slab temperature profile approaches a uniform distribution (Figure 10.c).

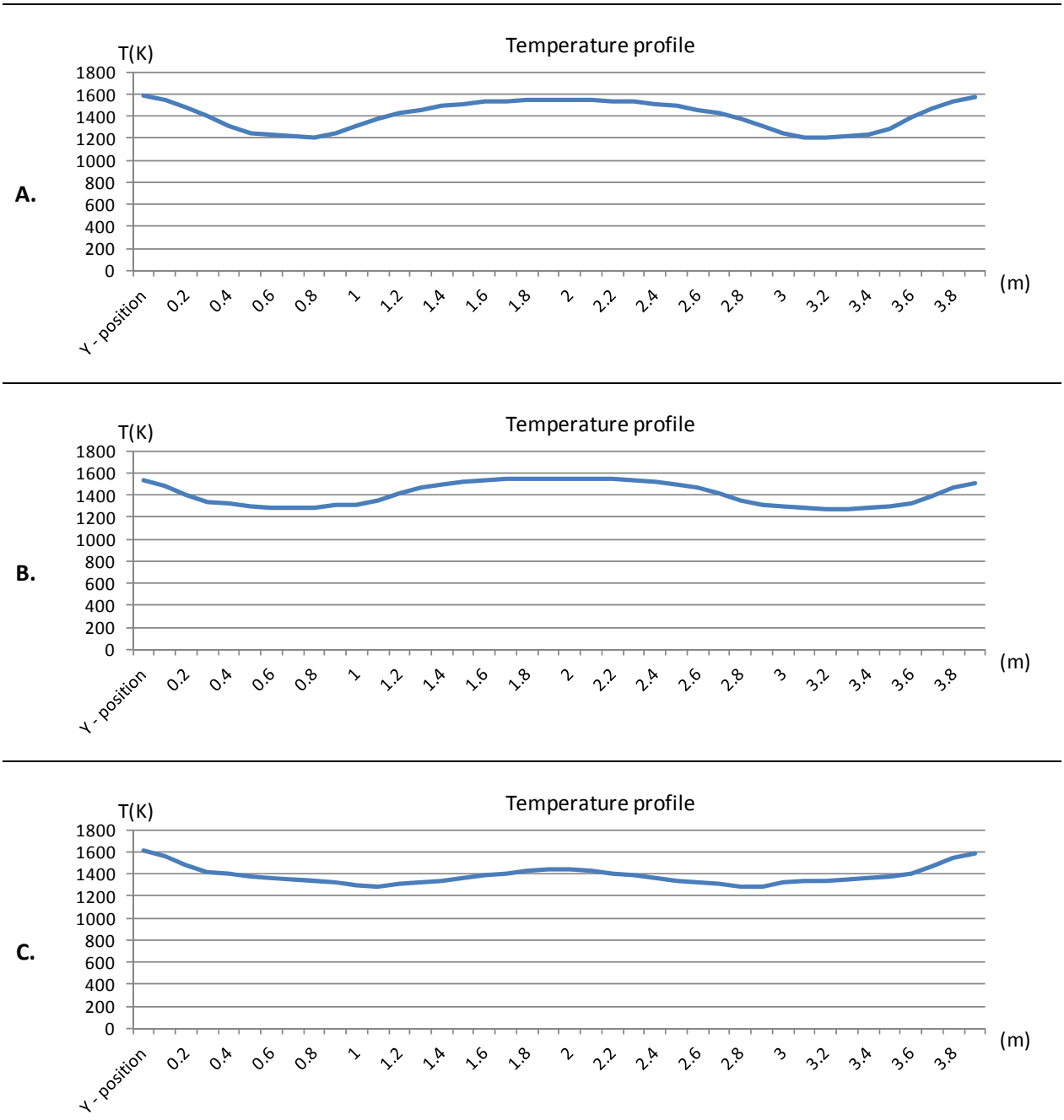


Figure 10: Comparison of the temperature profile at the end of the furnace for the different rail configuration cases (middle of the slab at 5cm height).

5.4.3 Heating temperature profile set point

Consider the set point temperature profile at the center of a slab presented in figure 11. This profile corresponds to a delayed heating temperature profile that was chosen because it delivers high efficiency for energy consumption. In order to reach this temperature profile in a slab, a specific wall temperature profile has to be determined. In figure 12, we show the corresponding temperature profile of the wall for a slab of dimensions 80 cm × 400 cm × 20 cm.

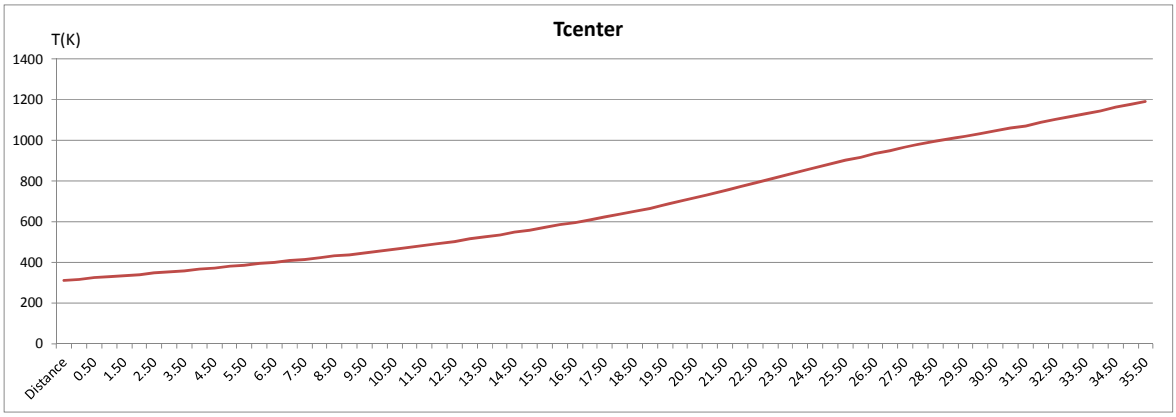


Figure 11: Heating temperature profile set point at the center of the slab

In this case, a trial and error approach is used for computing the wall temperature profile. Nevertheless, an inline process should continuously determine the walls temperatures in order to make the temperature profiles in all the slabs approach the set point temperature profile. So the possibility of predictive computing of the slab temperature profile during the heating process, demonstrated in paragraph 5.2.4, and the precision of the computations are extremely appealing for building a high accurate inline controller.

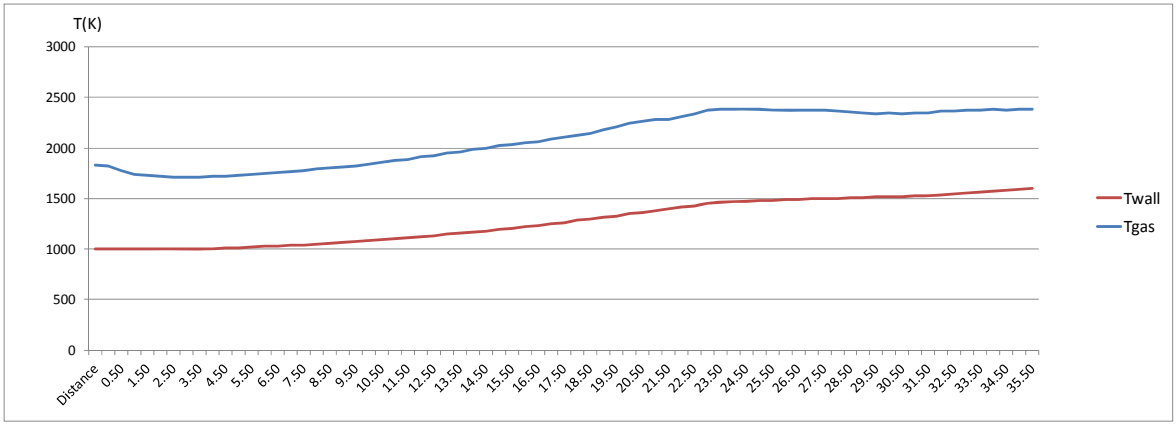


Figure 12: Temperature profiles as function of furnace length.

5.5 CONCLUSIONS AND FUTURE WORK

A global solution for computing and coupling the radiation and diffusion heat exchanges in a steel reheating furnace is first presented. The multi-grid approach is used for computing the direct exchange factors. The computation of the insignificant direct exchange factors is eliminated in order to avoid time waste. The total exchange factors are then computed by the NRPA while dividing the furnace into zones. The zone-by-zone computation implies the application of the NPRA to lower size systems, which achieves smaller computation time with small loss of accuracy. The radiation is then coupled to heat diffusion, which is computed using the efficient LOD scheme presented in chapter 4. As a result, a high precision 3D numerical solver is set and a very fast computation time is achieved that allows the pre-computation of the heat transfers for the next 6-7

minutes using an NVIDIA Tesla C 1060 GPU with 240 cores in 10 seconds. Moreover, the next two hours could be pre-computed using two NVIDIA GeForce GTX 690 GPUs with 3072 cores each. The computation throughput will still be increasing with new GPU generations, since their performance and number of cores are roughly doubling with each new generation. An example of thermal analysis model for a steel reheating furnace was presented. In this model, the wall temperature was taken as an input for computations and the temperatures of the walls and the gases filling the furnace were assumed to be one-dimensional in the direction of the length of the furnace. Then a black point analysis and a corresponding efficient configuration for rails positions were presented. Finally, a temperature profile for the walls was determined in order to achieve a precise delayed heating temperature at the center of a slab.

The high precision and computation speed that were achieved for the computations will be very useful for building highly accurate inline controllers in future work. It should be able to regulate wall temperature profiles in order to achieve precise heating temperature profiles and precise temperature profiles in the slabs at the end of the furnace. In this work, the wall temperature was assumed one-dimensional for the thermal model. Nevertheless, the numerical solution could be similarly applied for solving thermal model with two or three dimensional wall temperature profile without any loss of performance. Consequently, using the 3D heat diffusion solution, it could be possible to regulate the furnace in order to get precise heating temperature profiles in many points of the slabs.

Conclusions and Perspectives

In thesis, a highly efficient solution for the heat transfers in steel reheating furnaces was given, based on efficient solutions for the radiation heat transfer in the furnace and the heat diffusion in the steel slabs that are heated through the furnace.

The solution for computing radiation heat exchange distinguishes two main steps, first the computation of the direct exchange factors and then the deduction of total exchange factors from them based on the material physical properties.

The multiple absorption coefficient zonal method (MACZM) was considered as a method for computing direct exchange factors in semi-transparent media based on uniform space voxelization. At first, the method was combined together with a set of mean beam lengths based on neural network correlations that replaces pre-computed tables. The accuracy of MACZM was validated using an elementary numerical validation and an experimental validation on a flameless oxidation burners test furnace. On the other hand, it was compared to another efficient computational method, the flux planes approximation method. At the end, the computational efficiency of MACZM is highlighted.

A fast implementation of the Multiple Absorption Coefficient Zonal Method (MACZM) in 3D has then been presented. Efficient methods and algorithms have been discussed for the main parts of the MACZM implementation. 3D scan-conversion algorithms are used for the voxelization part, thus guaranteeing linear computation complexity in terms of mesh number. The second part is the most time consuming and it serves for computing a mean absorption coefficient over ray traversals. Taking into account topological constraints, a comparison of best discrete line algorithms has led to adapt the 6-parametric line algorithm for the computation of this part of MACZM.

A CUDA implementation of the method has then been described. The method was demonstrated to be massively parallel in a SIMD mode. An efficient solution for the optimization between the method parallelism (threading) and the use of the memory resources available on the GPU was discussed. Speed-ups achieved on the GPU range from 300 to 600 times, using a NVIDIA Tesla C 1060 GPU and an Intel Xeon CPU E5507 at 2.27 GHz.

Lastly, the computation of direct exchange factors inside a steel reheating furnace were considered and a multi-grid approach for MACZM has been presented in order to minimize the computation times while keeping good accuracy level for all radiative heat-exchange factors. A computation time of few seconds were achieved using the GPU implementation with an average speed-up of 320 times.

After that, the solution for computing total exchange factors from direct exchange factors was presented. In this purpose, the plating algorithm (PA) was first reviewed and CPU and GPU parallelization of this algorithm was presented. GPU parallelization of the PA give about 30 times

speed-up by comparison to the CPU serial PA run. The non-recursive plating algorithm NRPA was then presented in the aim to provide an algorithm for computing TEAs with lower computational complexity than the $O(N^3)$ complexity of the PA. The NRPA allows the computation of TEAs from DEAs in one single computation step. The formulations of the NRPA were identified from the PA. In this formulation an approximation was introduced consisting of neglecting all the terms containing the multiplication of more than m direct exchange areas for an NRPA of order m. The NRPA was then expressed in matrix form that consists mainly of a matrix multiplication. A review of matrix multiplication optimal algorithms helped demonstrating the computational complexity of the NRPA to be $O(N^{2.38})$ to the number of multiplication operations.

Efficient CPU and GPU implementations of the NRPA were then implemented based on the optimized library of linear algebra BLAS and its GPU CUDA version cuBLAS. Consequently, NRPA was found to run up to 70 times faster than PA on CPU. Moreover, the NRPA CUDA implementation is found to run up to 750 times faster than the PA serial CPU code. On the other hand, NRPA accuracy was tested on a test enclosure by comparison to the PA. NRPA accuracy is found to decrease with increasing physical reflectivity of the materials in the enclosure. Nevertheless, it achieved more than 99% accuracy with enclosure surfaces reflectivity of 0.75.

The application of the NRPA to the steel reheating furnace required the division of the computation into zones in order to achieve computation time similar to the computation time of the direct exchange areas. A small error of 1 or 2% is thus added to the computed total exchange areas.

An efficient GPU solution for the three-dimensional heat diffusion PDE based on finite difference split methods was then presented. Two parallel implementations of the LOD and ADI methods, an utmost parallelization and a stencil only parallelization, are then presented and compared. The utmost parallelization is new it allows maximum parallelization and is only applicable to the LOD method. It was implemented with both cyclic reduction CR and parallel cyclic reduction PCR solvers. On the other hand, the stencil parallelization is applicable to LOD method and ADI methods. Stencil parallelization was implemented with Thomas algorithm, in addition to CR and PCR. The LOD method was found to give fastest execution time between the split methods. On the other hand, utmost parallelization together with PCR solver is faster applied to small grids while stencil parallelization together with Thomas algorithm outperforms other algorithms for large grid size. Finally, an optimal solver was discussed based on a combined alternated solution scheme based on the utmost parallelization together with PCR solver at one side and the stencil parallelization together with Thomas algorithm on the other side. Compared to CPU implementation, the optimized CUDA implementation achieves high accelerations, even with small grid size. Finally, testing the LOD accuracy showed that the method allows large time steps while still delivering good precision.

The work is concluded by an analysis of black points in steel reheating furnaces resulting from rails cooling on steel slabs when leaving the steel reheating furnace. A 3D dynamic simulation of the reheating furnace was thus presented based on the coupling of the solutions discussed earlier. The thermal model of the furnace was computed based on the furnace wall temperature profile input. An efficient configuration for the rails positions was then given based on the 3D temperature profiles of the slabs at the end of the furnace.

The speed and the precision of the proposed solution for heat transfer in the steel reheating furnace will allow highly efficient inline control of these furnaces. At the first hand, it allows real-time simulation and highly accurate 3D temperature profiles to be computed in a furnace. On the second hand, the fastness of the solution for heat diffusion in slabs allows predictive computation to be done in parallel to the real-time dynamic simulation for simulating the 3D temperature profiles in the slabs. This allows better control of the furnace temperatures in order to reach precise set temperature profiles of the slabs. Besides, the computation speed results in possible optimization of highly efficient steel reheating furnaces. As well, it could be used for determining radiative properties of the surfaces and the gases in a furnace.

The developed methods are general and thus they could be applied for any applications that require computations similar to the computations in steel reheating furnace. It could be used for enhancing precision of a single simulation and deliver better understanding of the heat exchanges as well as it could be used for building efficient inline control processes that guarantee energy efficiency and quality enhancement. In steel reheating furnace for example, much finer mesh is now possible to be applied for the computations and temperature profiles are computed in 3D. Moreover the computation speed allows the heat process to be simulated numerically inline before it has been processed leading to new efficient control strategy.

The accuracy of the methods for computing radiation heat transfer in semi-transparent media and their computational efficiency make them of special interest for application to problems with semi-transparent media. Actually, the MACZM were first developed for computing heat exchange in semi-transparent media. In addition, the speed-up due its parallel execution on the GPU is higher for computing heat exchange factors between volumes than between surfaces. The methods are then highly suitable for applications that requires intensive computation in semi-transparent media as flame simulation, vapor explosion, optimizing problem configurations etc. On the other hand, the short computation time of the NRPA not only makes it very useful for application with MACZM but for applications that requires repetitive computation of total exchange factors without repeating the computation of direct exchange factors as determining gas or material properties.

Finally, the performance of the solution that is presented will be enhanced with every new GPU generation. This is because GPUs are parallel processors. Thus, their number of cores and then their computation capability is growing up with each new GPU generation.

GPU CUDA Programming

A

APPENDIX CONTENTS

A.1 Introduction 138
A.2 Cuda Program Structure 139
A.3 Cuda Memories 140
A.4 Cuda Threads 141
A.5 Performance Considerations 142

A.1 INTRODUCTION

CUDA or Compute Unified Device Architecture is a minimal C extension that allows parallel programming of GPUs; It does not recommend any graphics API knowledge. Figure 1 shows the architecture of a CUDA-capable GPU, the NVIDIA Tesla C 1060 (used in this work). A GPU is presented as a set of highly threaded streaming multiprocessors (SMs). Each SM has a number of streaming processors SPs with a cached shared memory. A GPU comes with up to 8 GB of graphics double rate (GDDR) DRAM, referred to as global memory. Thus, GPU DRAM is longer latency memory than CPU DRAM. The longer latency of GPU DRAM is hidden by higher memory Bandwidth and by running thousands of threads concurrently. All the parallel threads are executed on SPs, each SP is massively threaded and has a multiply-add (MAD) unit and an additional multiply unit. The Tesla C 1060 has 30 SM (8 SPs for each) and it supports up to 512 threads per SM which sums up to 15360 threads. Each CUDA-capable GPU is equipped with a hardware control unit for managing thread execution.

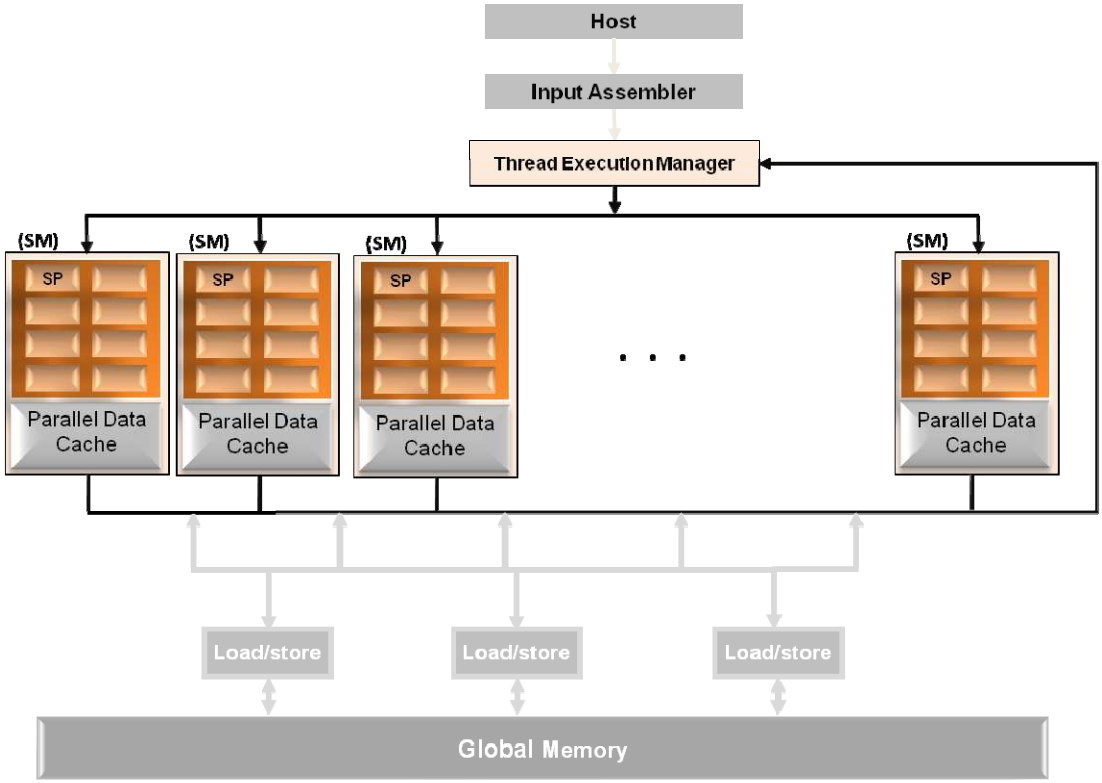


Figure 1: Architecture of a CUDA capable GPU.

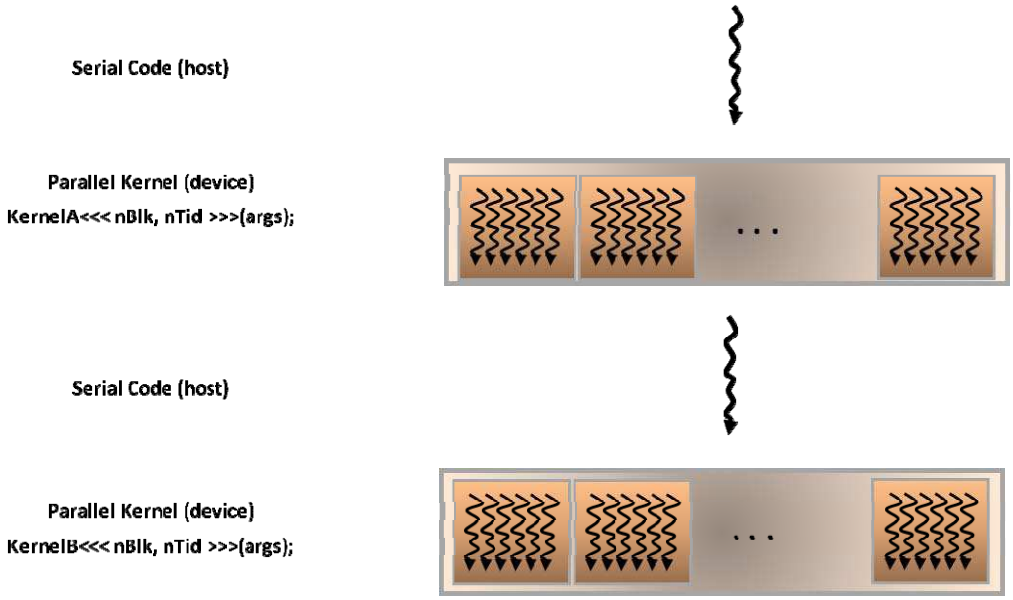


Figure 2: Typical kernel execution sequence in CUDA.

A.2 CUDA program structure

A typical CUDA program is a succession of CPU (host) and GPU (device) phases implemented in a single source code. In host phases are implemented the parts of the program with low or no data parallelism. In a device (GPU) phase (or kernel) is implemented a part of the program with high data parallelism. Figure 2 illustrates the execution of a typical CUDA program. The program always starts with a CPU phase (serial code). The kernel's initialization and its invocation are implemented in serial code using the statement:

$$Kernel <<< nblocks, nthreads >>> (args)$$

The kernel executes on the GPU and thus a large number of parallel threads are generated on the GPU. All the parallel threads in a kernel form a grid and the grid (figure 1). Hundreds and thousands of threads are needed to take advantage of abundant data parallelism and consequently to guarantee an efficient GPU program. Parallel threads take only few cycles to generate and schedule on the GPU, due to efficient hardware support.

Furthermore a kernel function is declared as a regular C function preceded by a CUDA specific key-word (ex: __global__) in front of it. The same kernel is executed by all the parallel threads but using different data input in each thread. CUDA programming is then a case of single-program, multiple-data parallel programming style [20]. Before executing any kernel on a data, this data have to be transferred from the host memory to the device memory.

A.3 Cuda memories

Figure 3 shows the CUDA device memory model. As seen in the figure, in CUDA each device has its own set of memory. A set of application programming interface (API) functions is provided by CUDA runtime in order to manage memory operations; Allocate and free memory on the device, transfer data between the host memory and the device memory, read and write memory on the device. The global memory and constant memory are device memories accessible from the host to transfer data from the host to the device or in the opposite way from the device to the host. The device can also read and write global memory and read only constant memory. These two memories are accessible by all the threads in the grid (all the threads in a kernel). Note here that a grid is divided into smaller groups of threads, the blocks. Each block has its shared memory that is accessible by all the threads in the block in read and write mode. Finally, each thread has its own set of registers. A register is a memory space containing only one variable at a time.

The shared memory is a very limited memory resource (16 KB or 48 KB) and the number of registers is limited (16384 in a NVIDIA Tesla C 1060 GPU). The constant memory is also very limited (64 KB). The global memory is DRAM and can reach up to 8GB.

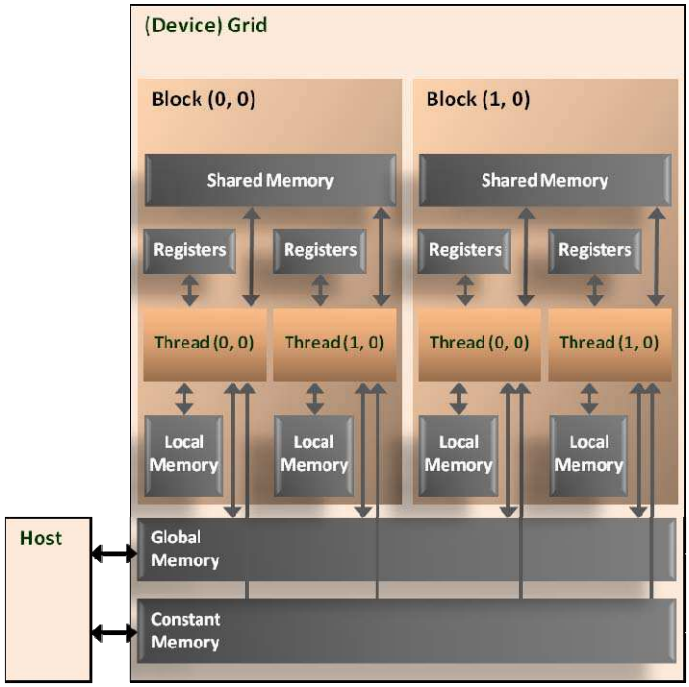


Figure 3: CUDA device memory model.

A.4 CUDA threads

As shown in figure 4, a CUDA grid composed of a 2D matrix of blocks and each block is a 3D matrix of threads. Blocks and threads are identified by their block ids $(bIdx, bIdy)$ and thread ids $(tIdx, tIdy, tidz)$. The number of the grid blocks can vary in the interval $[1, 65535]$ in each of the two dimensions. The number of threads in a block is limited (ex. 512 in a NVIDIA Tesla C 1060 GPU). When a kernel is lunched the number of blocks in the grid (gridDimx,gridDimy) and the number threads in each block$(blockDimx, blockDimy, blockDimz)$ have to be defined. Blocks can execute in any order. Each block of a grid can execute on only one SP. The number of blocks executing concurrently in the same SP is limited (8 for the Tesla C 1060).

The number of threads can vary in the interval $[1, limit]$ (ex. limit = 512) in each of the three block dimensions with the condition that total number of threads in the block does not exceed the limit. Threads execute in groups called tiles and warps (16 and 32 threads). All the threads in a warp execute the same instruction at a time. This style of execution is called single-instruction, multiple-thread (SIMD).

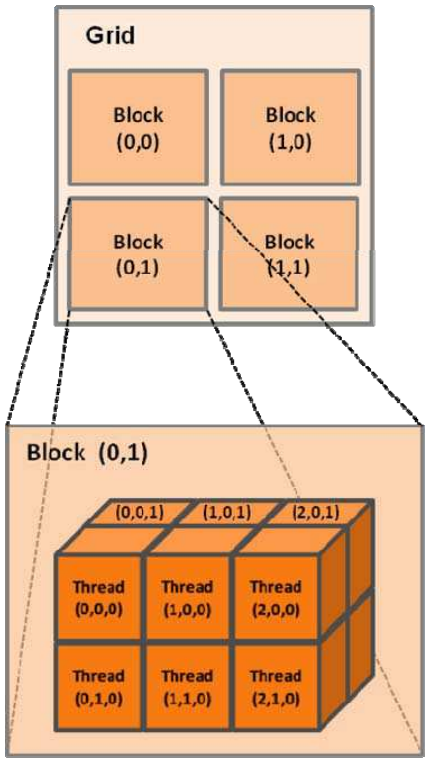


Figure 4: CUDA grid and block configuration.

A.5 Performance considerations

In order to achieve maximum performance on a GPU we need to maximize the number of grid threads executing concurrently on the GPU on the first hand. The thread divergence is when not all the threads of a warp execute the same instruction. In this case, the number of the warp threads is reduced and some threads have to wait for the others to finish so they continue their execution. Thus avoiding thread divergence when possible is crucial. An if-else instruction can result in thread divergence when its decision is not the same for all the threads in the warp, due to the SIMD execution. Divergence can also result when a loop of the kernel has a condition that does not achieve the loop at the same time in all the threads.

Since each block allocates its part of shared memory and registers (depending on its number of threads), the number of blocks in a SM is limited by the memory space. For example if we consider a G80 NVIDIA GPU with 8 SMs and a total shared memory of 16 KB, this gives 2KB per SM. In this case, if we use 1 KB of shared memory per block we can have at maximum two blocks running simultaneously on one SM. On the other hand highly diminishing the number of threads in a block could highly diminishes the number of threads executing on a SM since each SM can contain maximum eight blocks at a time. If we consider the previous example, having 2 blocks with 16 threads each on a SM gives only 32 threads executing simultaneously on the SM while it can support 768 threads. Therefore, the number of threads in a block has to be optimized for maximum performance according to the GPU characteristics.

On a second hand, performance grows up with higher ratio of instructions to the number of memory accesses (Read or write). In another manner, in the GPU, instructions are fast and memory accesses can be too long. The access to global memory takes up to 600 clock cycles while in a NVIDIA G80 GPU an instruction takes only 4 clock cycles. Fortunately, there is many ways to encounter memory latency. Global memory can be coalesced, this is when a block of neighbor variables are accessed once all together (16 Bytes of memory), and it takes the same time as for accessing one variable in the block. Shared memory is a low latency access memory. Another strategy for reducing global memory latency is to load memory by chunks and store it in shared memory while executing blocks. Then all the threads in a block can access the shared memory of the block. This is efficient when a variable is loaded once and used by many threads. Constant memory is stored in global memory but it is cached so when all threads access the same variable it is very fast. Constant memory is usually used to store data that will be read by the threads while executing the kernel. Finally, registers are very fast and threads use them for storing their regularly accessed variables.

RESUME DE LA THESE EN FRANCAIS

B

APPENDIX CONTENTS

B.1	**Calcul des Facteurs de Transferts Radiatifs Directs par MACZM**	**144**
B.2	**Implémentation et Parallélisation de MACZM**	**152**
B.3	**Calcul des Facteurs de Transferts Radiatifs totaux**	**162**
B.4	**Modélisation de La Diffusion 3D par les méthodes des différences finies**	**169**
B.5	**Application à la Simulation Ultra-Rapide d'un Four de Réchauffage Sidérurgique**	**182**

B.1 Calcul des Facteurs de Transferts Radiatifs Directs par MACZM

B.1.1 Introduction

Le calcul des facteurs de transferts radiatifs est généralement la principale difficulté dans la modélisation dynamique des transferts de chaleur dans les milieux à haute température. L'intégration directe de l'équation de transferts radiatifs est très difficile, même pour les applications les plus simples. Par conséquent, plusieurs méthodes numériques ont été développées. La plupart de ces méthodes restent très lentes ou peu précises, surtout en milieux participants et/ou multidimensionnels.

La méthode zonale à coefficients d'absorption multiples (MACZM) est basée sur le concept de facteur d'échange générique (GEF) similairement à la méthode zonale traditionnelle [1-3]. Cette méthode a été d'abord développée par Yuen [4], qui a introduit une nouvelle définition des GEF *somme* qui sont la superposition de GEF partiels. Ces derniers permettent de prendre en considération des changements brusques des propriétés radiatives du milieu, surtout autour des volumes émetteurs et récepteurs du rayonnement. La méthode est très efficace dans les milieux non-isothermes et non-uniformes.

Dans un travail plus récent, Yuen [5] a défini un ensemble de longueurs de faisceaux permettant de remplacer la tabulation des GEF par des corrélations de réseaux de neurones. Afin de valider l'approche, on considère le cas d'un four de réchauffage sidérurgique de brames, contenant un mélange de fumées qui rend le volume semi-transparent. D'autre part, on considère un outil efficace, MODRAY, développé par El Khoury [6]. MODRAY est basé sur la méthode des flux plans et conçu pour le calcul des facteurs de transferts radiatifs dans les enceintes fermées. Les facteurs de transferts radiatifs directs (ne prenant pas en compte l'effet des multi-réflexions du rayonnement) dans le four sont ensuite calculés par les deux méthodes simultanément et les valeurs sont comparées.

B.1.2 Méthode zonale à coefficients d'absorption multiples

Facteurs d'échanges directs

Un facteur d'échange générique (GEF), est défini comme étant la quantité de rayonnement émise par un élément de volume (ou voxel) ou un élément de surface et absorbée par un autre élément de volume ou de surface. Yuen [4] a défini trois facteurs d'échanges génériques partiels à deux composants chacun. La superposition des GEF partiels permet d'obtenir les GEF. Un maillage volumique, dont les voxels sont cubiques, est considéré, dans un repère de coordonnées cartésiennes (O, x, y, z). La position d'un voxel est déterminée par les coordonnées discrètes de son sommet inférieur gauche (figure 1).

L'échange radiatif entre deux éléments de volume est caractérisé par neuf GEF partiels. Deux cas de référence sont à distinguer, $(\mathrm{g_1 g_2})_{\mathrm{pp}}$ et $(g_1 g_2)_{pd}$ (figure 1). $(g_1 g_2)_{pp}$ est la fraction de l'échange radiatif entre les deux volumes qui est émise par V_1 à travers sa surface supérieure, et reçue par V_2 à travers sa surface inférieure, et $(g_1 g_2)_{pd}$ est la fraction échangée entre les deux

volumes à travers la surface de V_1 perpendiculaire à l'axe des "x" et la surface inférieure de V_2. Les GEF partiels, volume-volume, sont définis en fonction des trois épaisseurs optiques a_1D, a_2D et a_mD et des distances adimensionnelles (n_x, n_y, n_z) entre les deux voxels :

$$\frac{(g_1g_2)_{pp}}{D^2} = F_{ggpp}(a_1D, a_2D, a_{m,zz}D, n_x, n_y, n_z) \quad (1)$$

$$\frac{(g_1g_2)_{pd}}{D^2} = F_{ggpd}(a_1D, a_2D, a_{m,xz}D, n_x, n_y, n_z) \quad (2)$$

où D est la dimension caractéristique des voxels, a_1 et a_2 sont les coefficients d'absorption moyens dans V_1 et V_2 respectivement, a_m est l'absorptivité moyenne entre les centres des deux surfaces en considération (figure 1). a_m est calculé de proche en proche sur la ligne de centre à centre entre les deux surfaces en détectant tous les voxels traversés par la ligne et sommant la longueur de traversée de la ligne de chacun de ces voxels L_i, multipliée par le coefficient d'absorption moyen a_i associé à ce voxel (figure 2). La somme est ensuite divisée par la longueur totale L_c de la ligne joignant les deux centres : $a_m = (\sum a_iL_i)/L_c$. Le facteur d'échange entre deux voxels est obtenu par la superposition de 3^2 facteurs d'échanges génériques partiels volume-volume, étant donné qu'entre deux voxels, chaque voxel peut échanger du rayonnement avec l'autre à travers trois de ses facettes. Les sept autres GEF partiels sont obtenus à partir des deux cas de référence par des transformations

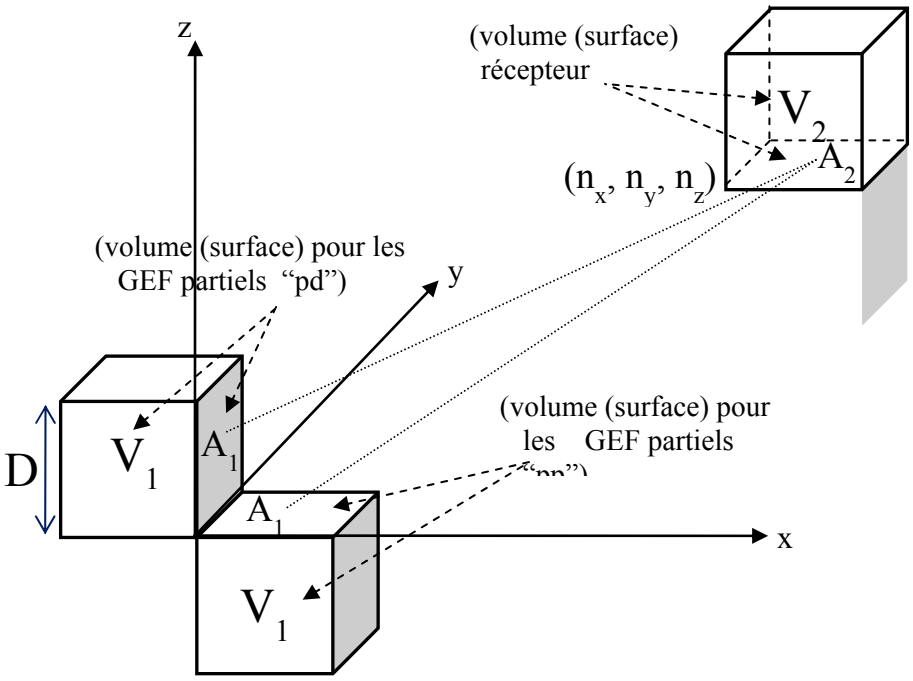


Figure 1 : Géométrie des GEFs partiels.

géométriques simples.

Le GEF partiel volume-surface à deux composantes $(g_1s_2)_{pp}$ et $(g_1s_2)_{pd}$ est défini de la même manière en fonction de a_1D et a_mD et(n_x, n_y, n_z). De même, pour le GEF surface-surface, qui est défini uniquement en fonction de a_mD et(n_x, n_y, n_z).

Le facteur d'échange direct entre deux objets de l'espace discret est égal à la somme de tous les facteurs d'échanges génériques entre leurs voxels. Yuen [4] tabule les facteurs d'échanges génériques partiels dans des tableaux utilisés pour l'application de MACZM. Cependant, ceci

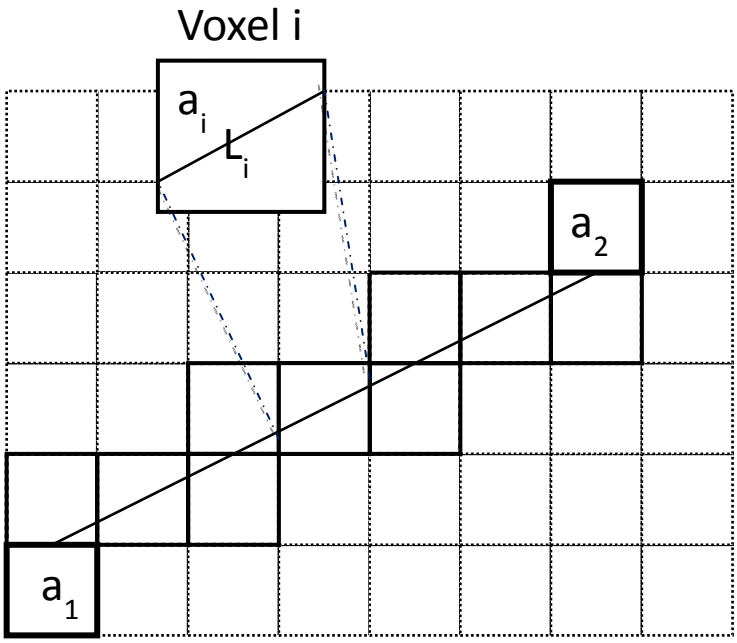


Figure 2 : Calcul de a_m en 2D

nécessite un grand espace mémoire et engendre une petite erreur inévitable due aux interpolations lors des lectures à partir des tableaux.

Longueurs moyennes de faisceaux de neurones

Yuen [5] a défini un ensemble de longueurs moyennes de faisceaux (MBL), pour exprimer les facteurs d'échanges génériques partiels dans des équations monodimensionnelles. Il a démontré que six MBL sont suffisantes pour la modélisation des transferts radiatifs tridimensionnels entre volumes, volumes et surfaces, et entre surfaces. Celles-ci sont divisées en trois catégories, les MBL de transmission, d'émission et d'absorption. En cohérence avec les GEF partiels, les longueurs moyennes de faisceaux sont aussi définies en fonction des mêmes épaisseurs optiques $a_1 D$, $a_2 D$ et $a_m D$ et distances adimensionnelles (n_x, n_y, n_z).

La longueur moyenne de faisceau de transmission est physiquement équivalente à la longueur de trajet monodimensionnel qui permet de transmettre la même quantité de rayonnement entre deux éléments de surface dans le milieu considéré. Elle est reliée aux facteurs d'échanges génériques surface-surface par les équations suivantes :

$$\frac{(s_1 s_2)_{pp}}{D^2} = F_{12,pp}(n_x, n_y, n_z) e^{-a_m L_{t,pp}} \quad (3)$$

$$\frac{(s_1 s_2)_{pd}}{D^2} = F_{12,pd}(n_x, n_y, n_z) e^{-a_m L_{t,pd}} \quad (4)$$

où $L_{t,pp}$ et $L_{t,pd}$ sont les deux longueurs moyennes de faisceau de transmission. $F_{12,pp}$ et $F_{12,pd}$ sont les facteurs de formes entre les deux éléments de surface.

De même, la longueur moyenne de faisceau d'émission est interprétée physiquement comme étant la longueur de trajet monodimensionnel permettant d'avoir la même émissivité que le voxel émetteur. L'extension des équations précédentes donne les relations entre la longueur moyenne de faisceau d'émission et les facteurs d'échanges génériques volume-surface,

$$\frac{(g_1 s_2)_{pp}}{D^2} = F_{12,pp}(n_x, n_y, n_z)(1 - e^{-a_1 L_{em,pp}})e^{-a_m L_{t,pp}} \quad (5)$$

$$\frac{(g_1 s_2)_{pd}}{D^2} = F_{12,pd}(n_x, n_y, n_z)(1 - e^{-a_1 L_{em,pd}})e^{-a_m L_{t,pd}} \quad (6)$$

où $L_{em,pp}$ et $L_{em,pd}$ sont les deux longueurs moyennes de faisceaux d'émission.

La longueur moyenne de faisceau d'absorption est interprétée physiquement comme étant la longueur de trajet monodimensionnel permettant d'avoir la même absorptivité que le voxel absorbant. De même, les relations entre la longueur moyenne de faisceau d'absorption et les facteurs d'échanges génériques volume-volume sont données par :

$$\frac{(g_1 g_2)_{pp}}{D^2} = F_{12,pp}(n_x, n_y, n_z)(1 - e^{-a_2 L_{a,pp}})(1 - e^{-a_1 L_{em,pp}})e^{-a_m L_{t,pp}} \quad (7)$$

$$\frac{(g_1 g_2)_{pd}}{D^2} = F_{12,pd}(n_x, n_y, n_z)(1 - e^{-a_2 L_{a,pd}})(1 - e^{-a_1 L_{em,pd}})e^{-a_m L_{t,pd}} \quad (8)$$

$L_{a,pp}$ et $L_{a,pd}$ étant les deux longueurs moyennes de faisceaux d'absorption.

Compte tenu de la complexité du comportement des MBL en fonction de leurs variables d'entrées, elles sont estimées par des réseaux de neurones artificiels, à deux couches, générés par apprentissage. L'erreur engendrée par les réseaux de neurones est inférieure à 5 %. Les données utilisées pour l'apprentissage sont générées par intégration numérique directe et pour des intervalles limités de variables d'entrées ; les réseaux de neurones sont ensuite générés pour les mêmes intervalles. Les épaisseurs optiques $a_1 D$, $a_2 D$ et $a_m D$ varient dans l'intervalle entre 0,01 et 4,6 pour tous les cas. Les valeurs maximales des distances adimensionnelles varient légèrement pour chaque réseau de neurones, mais en restant proches de 8 dans tous les cas. Pour des épaisseurs optiques supérieures à 4,6, le volume s'approche d'un corps noir ; le facteur d'échange générique entre des voxels distants est suffisamment petit et ne contribue pas significativement dans le bilan total de transferts de chaleur. Dans le calcul, les épaisseurs optiques supérieures à 4,6 sont remplacées par la valeur 4,6 et les distances adimensionnelles dépassant les limites, sont remplacées par les plus proches qui sont à l'intérieur des limites et qui correspondent au même trajet. Plus de détails sur les réseaux de neurones peuvent être trouvés dans Yuen [5].

B.1.3 Calcul des facteurs de transferts radiatifs directs par MODRAY

MODRAY est un outil qui permet de calculer les facteurs de transferts radiatifs surface-surface, surface-volume et volume-volume. Il est basé sur la méthode des flux plans [7, 8] et les relations d'Emery et al. [9].

Calcul des facteurs de transferts radiatifs directs surface-surface $\overline{S_iS_j}$ par la méthode des flux plans

Dans la méthode des flux plans une formulation angulaire des facteurs de transferts radiatifs directs

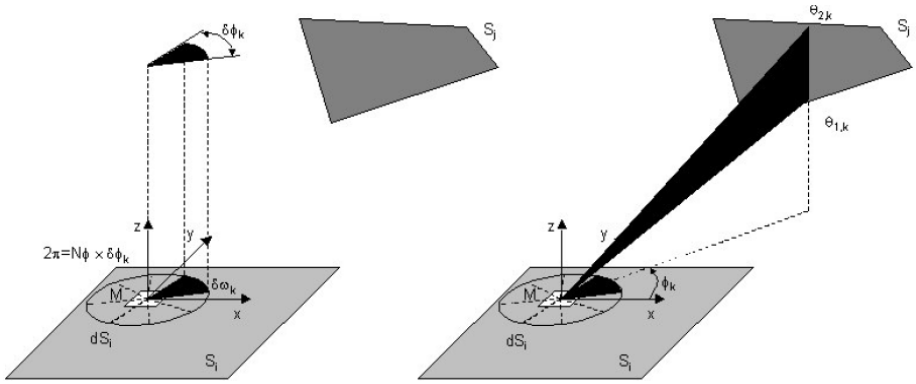


Figure 3 : Approximation des flux plans

est considérée. Les flux sortant d'une surface sont discrétisés en des quarts de plans appelés les flux plans, et les formules sont ensuite intégrées numériquement. La méthode induit une approximation appelée l'approximation des flux plans.

Considérons une surface S_i et un point M sur S_i, dS_i est un élément de surface de S_i entourant M (Figure 3). Vue du point M, une surface S_j se trouve dans la partie de l'espace située entre $\emptyset_1$ et $\emptyset_2$ en coordonnée angulaire $\emptyset$, et pour chaque valeur de $\emptyset$ entre θ_1 et θ_2. Dans un milieu absorbant gris non diffusant, le rayonnement reçu par S_j à partir de dS_i est calculé par :

$$dq_{ij} = \int_{\emptyset_1}^{\emptyset_2} \int_{\theta_1(\emptyset)}^{\theta_2(\emptyset)} I_i \, \tau_{ij} \sin\theta \cos\theta \, d\theta \, d\emptyset \tag{9}$$

I_i est la luminance de rayonnement isotropique émis par dS_iet τ_{ij} est la transmissivité moyenne entre dS_i et S_j le long du rayon de direction $(\emptyset, \theta)$.

Sachant que le flux total émis par dS_i dans toutes les directions est égal à πI, et après intégration sur S_i, la formulation angulaire du facteur de transferts radiatifs directs entre S_i et S_j est :

$$\overline{S_iS_j} = \frac{1}{2\pi} \int_{S_i} \int_{\emptyset_1}^{\emptyset_2} \int_{\theta_1(\emptyset)}^{\theta_2(\emptyset)} \tau_{ij} d(sin^2\theta) d\emptyset dS_i \tag{10}$$

L'hémisphère entourant M est divisée en un nombre fini $N_\emptyset$ de partitions, avec des angles solides $\delta\omega_k (k = 1, N_\emptyset)$ égaux. Chaque angle solide étant inclus entre deux quarts de plans perpendiculaires à S_i et faisant un angle $\delta\emptyset_k = 2\pi/N_\emptyset$ entre eux. Le flux radiatif dq_i, émis par dS_i, st ainsi égal à la somme des flux $dq_{i,k}$ émis du point M dans les différents angles de solide $\delta\omega_k (k = 1, N_\emptyset)$. L'approximation des flux plans consiste alors à considérer que chacun des flux $dq_{i,k}$ est concentré dans le plan bissecteur de $\delta\omega_k$. L'équation (10) devient :

$$\overline{s_i s_j} = \frac{1}{N_\emptyset} \int_{S_i} \left(\sum_{k:\, \emptyset_1 \le \emptyset_k \le \emptyset_2} \int_{\theta_1(\emptyset)}^{\theta_2(\emptyset)} \tau_{ij}\, d(sin^2\theta) dS_i \right) \quad (11)$$

$\theta_{1,k}$ et $\theta_{2,k}$ étant les limites de S_j dans le plan bissecteur. Les deux intégrales de l'équation (15) sont effectuées numériquement par une quadrature de Gauss.

Calcul des facteurs d'échanges radiatifs directs surface-volume $\overline{s_i g_j}$ et volume-volume $\overline{g_i g_j}$ par les relations de Emery et al.

Emery et al. ont démontré que les facteurs d'échanges radiatifs directs surface-volume et volume-volume peuvent être déduits des facteurs surface-surface par des considérations de conservation et compte tenu que chaque volume est entouré par un nombre fini de surfaces. L'approche détaillée est présentée par Emery et al. [9]. A titre d'exemple, si on considère un volume V_j entouré par les surfaces s_a, s_b, s_c et s_d dont s_a et s_b sont exposées à une surface S_i et s_c et s_d sont cachées par le volume V_j, alors le facteur d'échange direct entre V_j et S_i est :

$$\overline{s_i g_j} = \overline{s_i s_a} + \overline{s_i s_b} - \overline{s_i s_c} - \overline{s_i s_d} \quad (12)$$

B.1.4 Application à un four de réchauffage sidérurfique

On considère un four d'essais de réchauffage sidérurgique de brames d'acier. L'intérieur du four est rectangulaire (figure 4). Les dimensions internes du four sont $300\ cm \times 160\ cm \times 110\ cm$ et il est équipé de deux brûleurs sans flamme, placés dans la partie supérieure du four. Les deux brûleurs identiques sont modélisés par une étude CFD et sont démontrés équivalents à deux parallélépipèdes de dimensions $100\ cm \times 28\ cm \times 28\ cm$. Le four peut contenir une brame d'acier de dimensions $100\ cm \times 100\ cm \times 22\ cm$ et reposant sur quatre supports.

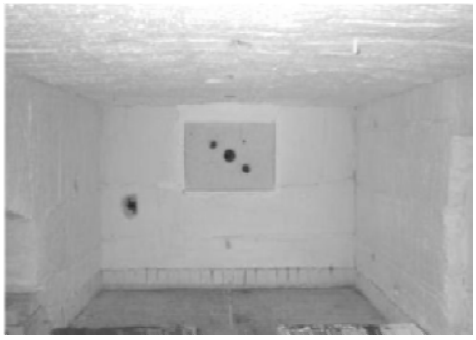


Figure 3 : Vue interne du four

La modélisation dynamique ainsi que les validations expérimentales des simulations sont détaillées par Ferrand [8]. Dans ce travail, on s'intéresse aux calculs des facteurs d'échanges radiatifs directs. Ceux-ci sont tout d'abord calculés par MACZM en appliquant un maillage de $30 \times 16 \times 11$ voxels (figure 5). Les facteurs d'échanges radiatifs directs sont ensuite générés dans MODRAY, ce qui engendre des maillages surfaciques. Trois cas de maillage sont considérés allant de $6 \times 6 \times 2$ mailles pour la brame dans le cas (a) jusqu'à $12 \times 12 \times 4$ dans le cas (c) qui correspond au maillage le plus fin considéré.

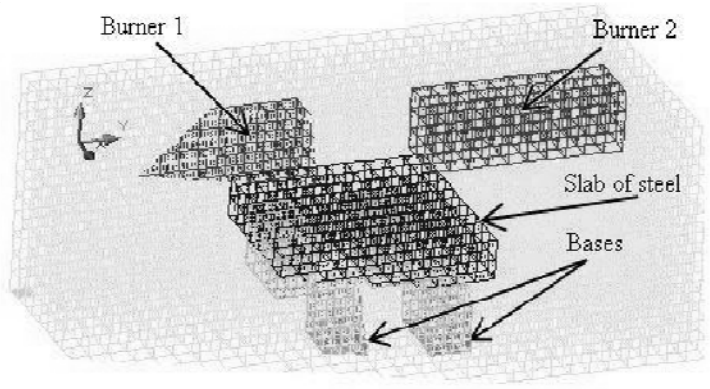


Figure 4 : Maillage du four (MACZM)

Les facteurs de transferts directs sont présentés tableau 1. Les résultats de MACZM présentent un bon agrément avec le cas (c) (le maillage le plus fin) pour MODRAY. MODRAY étant plus précis quand le maillage est plus fin, les résultats issus de MACZM sont ainsi validés. On note que les valeurs issues de MODRAY ont été validées expérimentalement par Ferrand [8]. De plus, on remarque que le temps de calcul total relatif à MACZM est très petit, de l'ordre de 2 s (comparé à 25 min pour MODRAY dans le cas (c)) ; ceci est obtenu grâce au concept de superposition de facteurs d'échanges génériques.

Tableau 1 : Facteurs de transferts radiatifs directs calculés par MACZM et MODRAY entre différents éléments du four de réchauffage sidérurgique considéré

Comparaison MACZM - MODRAY

Facteurs de transferts	MACZM	MODRAY Cas (a)	MODRAY Cas (b)	MODRAY Cas (c)
[Temps de calcul]	[2 s]	[25 min]	[6-7 min]	[2 min]
Brûleur (1) - Brûleur (2)	1,13560E-04	9,98015E-05	1,18587E-04	1,27992E-02
Brûleur (1) - Brame	1,03177E-02	1,00157E-02	9,76513E-03	8,68062E-02
Brûleur (2) - Brame	1,04588E-02	1,01281E-02	9,95802E-03	8,56366E-02

B.1.5 Conclusions et perspectives

La méthode zonale à coefficients d'absorption multiples a été programmée avec un ensemble de longueurs moyennes de faisceaux et des réseaux de neurones artificiels qui leur sont associés. D'autre part, l'outil MODRAY, basé sur la méthode des flux plans pour le calcul des facteurs de transferts radiatifs directs surface-surface, a été utilisé comme référence. MACZM est validée par comparaison à MODRAY sur le cas d'un four de réchauffage sidérurgique. Le temps de calcul relatif à la méthode est très court (2 s dans le cas du four). Ceci rend la méthode de grande importance pour la modélisation dynamique des facteurs de transferts radiatifs dans des systèmes mettant en jeu des milieux transparents ou semi-transparents.

Références

1. H. C. Hottel and A. F. Sarofim, Radiative Transfer, *McGaw Hill*, New York, 1967.
2. J. J. Nobel, The Zone Method: Explicit Matrix Relations for Total Exchange Areas, *Int. J. Heat Mass Transfer*, vol. 18, pp. 261-269, 1975.
3. W. W. Yuen and E. E. Takara, The Zonal Method: A Practical Solution Method for Radiative Transfer in Nonisothermal Inhomogeneous Media, *Annual Reviews of Heat Transfer*, vol. 8, pp. 153-215, 1997.
4. W. W. Yuen, The Multiple Absorption Coefficient Zonal Method (MACZM), an Efficient Computational Approach for the Analysis of Radiative Heat Transfer in Multidimensional Inhomogeneous Nongray Media, *Numerical Heat Transfer*, Part B, vol. 49, pp. 89-103, 2006.
5. W. W. Yuen, Definition and Evaluation of Mean Beam Lengths for Applications in Multidimensional Radiatve Heat Transfer: A Mathematically Self-Consistent Approach, *J. Heat transfer*, vol.130, 2008.
6. G. El Hitti, M. Nemer, K. El Khoury, and D. Clodic, Modified Zonal Method for Thin Semi-transparent Media with Reflective Boundary, *Proc. HT2007*, Vancouver, Canada, 32033, ASME-JSME, 2007.
7. G. El Hitti, M. Nemer, K. El Khoury, and D. Clodic, Transient Radiation Heat Transfer in Glass Sheets by the Thin Layer Approximation, *J. Heat transfer*, vol.132, 2010.
8. L. Ferrand, Modélisation et expérimentation des fours de réchauffage sidérurgiques équipés de brûleurs régénératifs à oxydation sans flamme, *Thèse, ENSMP*, Paris, 2003.
9. F. Emery, O. Johansson, A. Abrous, Radiation Heat Transfer Shapefactors for Combustion Systems, Fundamentals and Applications of Radiation Heat Transfer, *ASME HTD*, vol. 72, pp. 119-126, 1987.

B.2 Implémentation et Parallélisation de MACZM

B.2.1 Introduction

La modélisation des transferts de chaleurs radiatifs reste parmi les plus difficiles, surtout en milieux semi-transparents. Les fours de réchauffage sidérurgique sont un bon exemple d'application où l'échange de chaleur par rayonnement est dominant en raison des températures de fonctionnement élevées et avec un milieu non-homogène formé par les gaz de combustion circulant dans le four. Dans ce travail, un four de réchauffage d'acier est considéré. Des brames d'acier circulent lentement dans le four, où elles sont chauffées de la température ambiante à une température d'environ $1\,200°C$. A la sortie du four, les brames d'acier peuvent subir des procédés variés, dont par exemple le laminage. En général, la qualité du produit final est très sensible au profil de température dans les brames d'acier à la sortie du four de réchauffage. Un processus de contrôle dynamique en ligne permettant d'obtenir les profils de température souhaités à la sortie est donc d'importance majeure. De plus, il permettrait de réduire la consommation du four en optimisant les courbes de chauffe et en évitant les surchauffes. Cependant, il est impossible de se baser sur des mesures de température en temps réel dans le four à cause des températures très élevées, des fumées et du mouvement des brames d'acier dans celui-ci. Il faut donc simuler numériquement en dynamique les profils de température dans le four. Un modèle rapide pour la simulation des transferts radiatifs est donc nécessaire.

La méthode zonale à coefficients d'absorption multiples (MACZM) est une méthode récente publiée par Yuen [1]. Elle est basée sur le concept de facteurs d'échanges génériques (GEF). Yuen [1] a redéfini les GEFs comme étant la superposition de facteurs d'échanges génériques partiels afin d'augmenter l'efficacité et la précision de la méthode en milieu semi-transparent. En outre, le calcul est appuyé par l'utilisation de réseaux de neurones artificiels (RNA) [2,3]. Une étude démontrant la validité de la MACZM a été présentée dans Ghannam et al [3]. Dans ce travail, l'algorithme relatif à la méthode est présenté et une implémentation efficace de la méthode basée sur les algorithmes de géométrie discrète est également discutée.

En outre, la possibilité de parallélisation massive de la MACZM en programmant l'exécution sur les cartes graphiques (GPU) est démontrée. Les cartes graphiques (GPU) sont des processeurs à plusieurs cœurs de calcul qui ont été à l'origine conçus pour les jeux. Elles ont ensuite été utilisées pour la programmation à usage général dans le but de tirer profit de la puissance de calcul élevée qu'elles peuvent fournir. Néanmoins, une bonne connaissance des API graphiques était nécessaire et seuls quelques programmeurs spécialisés pouvaient développer des codes efficaces utilisant lesGPU. Ceci a changé avec l'apparition de CUDA en 2007 [4, 5] qui a augmenté la programmabilité des GPU et l'a rendue accessible à la communauté des scientifiques et des ingénieurs. CUDA est une extension du C, associé à un support matériel ajouté au GPU et permet la programmation efficace du GPU sans passer par les API graphiques. De nombreuses applications scientifiques (en particulier dans la photographie biologique) ont été développées avec CUDA et des performances élevées ont été atteintes. Dans ce travail, MACZM est programmée avec CUDA et des accélérations variant entre 300 et 600 fois sont obtenues par rapport au calcul séquentiel sur CPU.

Finalement, MACZM est appliquée pour la simulation d'un four de réchauffage de brames d'acier. Le four est représenté en détail, de même que la discrétisation et la résolution du four dans MACZM. Un maillage multi-grille est aussi présenté, permettant la réduction du temps de calcul en gardant un taux de précision élevé.

B.2.2 Méthode zonale à coefficients d'absorption multiples, algorithme et implementation

Formulation mathématique

La méthode zonale à coefficients d'absorption multiples MACZM [1-3] est basée sur le concept de facteur d'échange générique (GEF). Un facteur d'échange générique est la fraction d'énergie émise par un élément de volume ou de surface et absorbée par un autre élément de volume ou de surface (figure 1). L'espace est discrétisé en éléments de volume (voxels) cubiques. Des réseaux de neurones artificiels (RNA) sont générés pour permettre de calculer le GEF entre tout couple de voxels dans l'espace discret. Un GEF dépend de la distance entre les deux voxels, leurs absorptivités et la transmissivité du milieu qui les sépare. Cependant les RNA ne génèrent pas directement les GEF mais des entités physiques, les longueurs moyennes de faisceaux (MBL) qui sont liées aux GEF par des équations simples, comme suit :

$$\frac{g_1 g_2}{D^2} = F_{gg} = F_{12}(n_x, n_y, n_z)(1 - e^{-a_2 L_a})(1 - e^{-a_1 L_{em}})e^{-a_m L_t} \qquad (1)$$

où $g_1 g_2$ est le GEF entre deux éléments de volume, L_t, L_{em} and L_a sont respectivement les longueurs moyennes de faisceaux de transmission, d'emission, et d'absorption, $n_x, n_y, et\ n_z$ sont les distances adimensionnelles entre les deux voxels, F_{12} est un facteur de forme entre deux facettes des voxels et finalement a_m est le coefficient d'absorption moyen du milieu séparant les deux voxels.

Les longueurs moyennes de faisceaux sont des fonctions des mêmes variables que les GEFs et sont données à partir de celles-ci par les réseaux de neurones artificiels. Dans l'espace discrétisé, les distances adimensionnelles et le facteur de forme sont donnés par la géométrie. L'absorptivité d'un voxel correspond à l'absorptivité moyenne du volume occupé par ce voxel dans l'espace continu. Le seul paramètre restant est donc le coefficient d'absorption moyen a_m qu'il faut calculer pour chaque GEF.

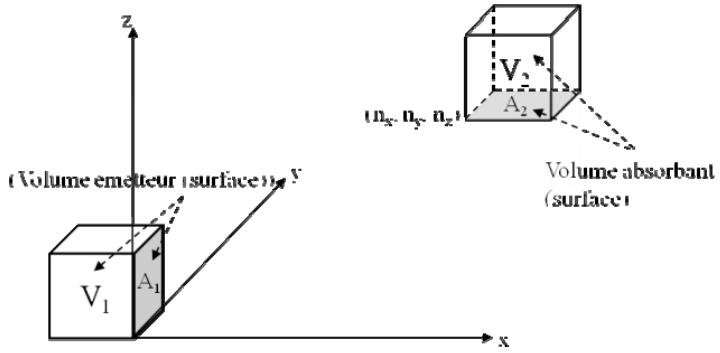


Figure 1 : Géométrie des GEFs

Calcul du coefficient d'absorption moyen

Comme indiqué précédemment, un maillage uniforme discrétisant les objets et les volumes de l'enceinte en des voxels cubiques est tout d'abord appliqué. Les objets sont discrétisés en se basant sur les algorithmes de "scan-conversion" [6]. Une étude de la connexité du milieu discret est présentée dans [7]. Le calcul de a_m se fait le long d'un rayon joignant les centres de deux facettes appartenant respectivement aux deux voxels considérés, comme suit :

$$a_m = \left(\sum_i a_i L_i\right) / L \tag{4}$$

où l'indice i réfère aux voxels traversés par la droite continue. a_i est le coefficient d'absorption d'un voxel i et L_i la distance traversée par la droite continue dans ce voxel.

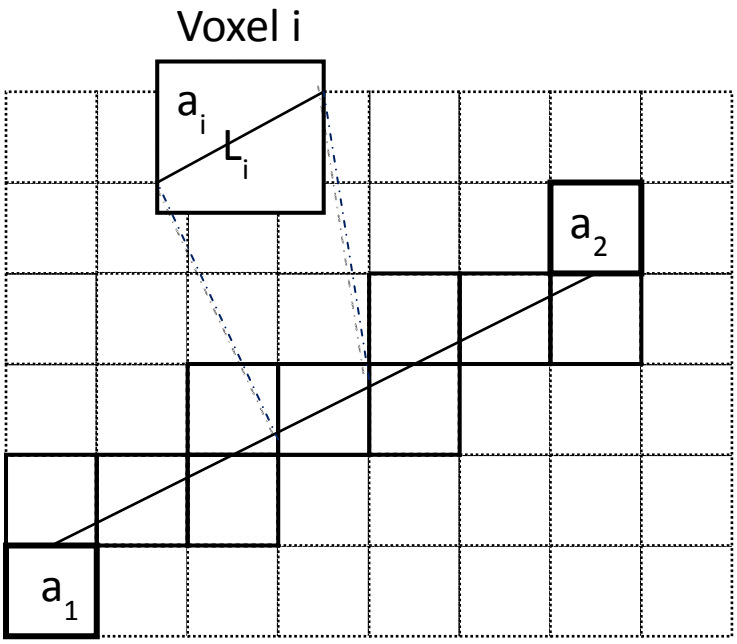


Figure 2 : *Calcul de a_m en 2D*

Par conséquent, le calcul de a_m nécessite de déterminer tous les voxels percés par la droite continue, et la longueur de traversée dans chacun (figure 2). Les algorithmes de droites discrètes de la littérature permettent de déterminer les voxels percés par la droite de manière efficace. Il reste à les modifier afin de calculer la distance traversée par la droite dans chaque voxel. Une étude comparative, permettant de déterminer l'algorithme de tracé de droites discrètes modifié le plus efficace en temps de calcul et prenant en compte les contraintes de connexité, est présentée dans Ghannam et al [7]. D'après cette étude, l'algorithme d'Amanatides et Woo [7] modifié est adopté.

Parallélisation de MACZM dans CUDA

Les cartes graphiques (GPU) exécutent les instructions en mode SIMT (Single-Instruction, Multiple-Thread), qui est un cas particulier du mode SPMD (Single-Program, Multiple-Data). Dans ce mode de parallélisation, plusieurs fils exécutent les mêmes instructions en parallèle mais sur des données différentes. Afin de prouver que l'implémentation de MACZM avec CUDA est faisable, il faut tout d'abord démontrer que MACZM est parallélisable en mode SIMT. Comme présenté dans

les paragraphes précédents, le calcul d'un GEF dans MACZM s'effectue en deux étapes principales. La première étape consiste à calculer le coefficient d'absorption moyen qui lui correspond. Ceci est effectué en utilisant un algorithme de droite discrète joignant les deux voxels qui correspondent au GEF calculé. Cet algorithme s'applique à n'importe quel couple de voxels indépendamment des autres. Il est ensuite exécutable en parallèle en mode SIMT. La seconde étape du calcul du GEF consiste à calculer les MBL par l'application des réseaux de neurones artificiels. Ces derniers sont les mêmes pour tous les GEFs, ce qui rend cette partie du calcul des GEF aussi parallélisable en mode SIMT. Le calcul des GEF est donc entièrement parallélisable sur GPU. En outre, 99,8 % du temps d'exécution sur CPU est dédié au calcul des GEFs. Le reste étant consacré à l'initialisation de l'enceinte et sa discrétisation, il sera caché par le temps de calcul sur GPU tout en étant exécuté sur CPU.

A partir de là, il reste à garantir un taux de parallélisation suffisant et effectuer une implémentation optimisée dans CUDA afin de tirer le maximum de performance du GPU. Une étude détaillée de l'implémentation de MACZM dans CUDA est présentée dans [7] et n'est pas répétée ici. En résumé, en utilisant un CPU Xeon E5507 à 2,27 GHz et un GPU NVIDIA Tesla C 1060 (240 cœurs), le calcul du coefficient d'absorption moyen s'exécute 150 fois plus rapidement sur GPU que sur CPU et l'application des réseaux de neurones artificiels se fait 400 à 1000 fois plus rapidement, selon la taille de la grille. Finalement, l'accélération de MACZM sur GPU avec CUDA varie entre 300 fois (GEF surface-surface) et 600 fois (GEF volume-volume).

B.2.3 Application à un four de réchauffage sidérurgique et approche multi-grille

Description du four

On considère un four de réchauffage de brames d'acier (figure 3). Les brames d'acier sont introduites dans le four à température ambiante. Elles sont positionnées sur quatre rails sur lesquels elles circulent lentement tout au long du four.

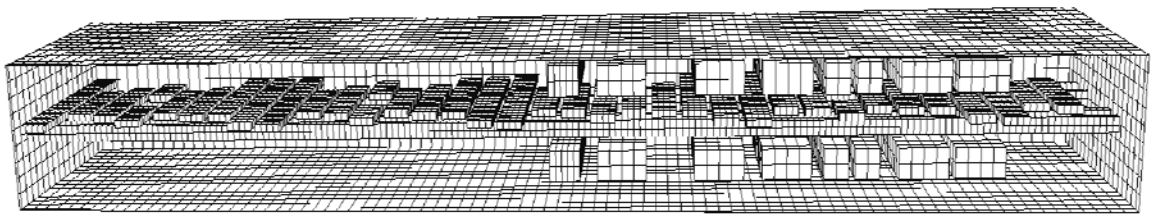


Figure 3 : Four de réchauffage de brames d'acier

Les dimensions internes du four sont 36,275 m × 6,8 m × 3,8 m. Il peut contenir jusqu'à 44 brames d'acier. Les brames d'acier ont des largeurs variables, mais leur hauteur et leur longueur sont constantes, égales à 0,2 m et 0,8 m respectivement. Une distance moyenne de 0,2 m sépare les brames d'acier dans le sens de la longueur du four. Un exemple de répartition des brames dans le four est représenté figure 4. Les rails sont supposés de section rectangulaire et leurs dimensions sont 36,275 m × 0,2 m × 0,4 m. Enfin, des brûleurs sans flamme sont positionnés dans les zones

supérieure et inférieure du four. Tous les brûleurs sont identiques et les volumes de combustion sont supposés parallélépipédiques, de dimensions 0,8 m × 4,0 m × 0,8 m.

Différents maillages seront considérés pour les simulations. Pour les maillages les plus fins, la simulation du four entier sera trop lente. Pour cela, les simulations seront effectuées tout d'abord sur un cas simplifié (figure 4) qui consiste en une tranche du four de longueur 7,2 m.

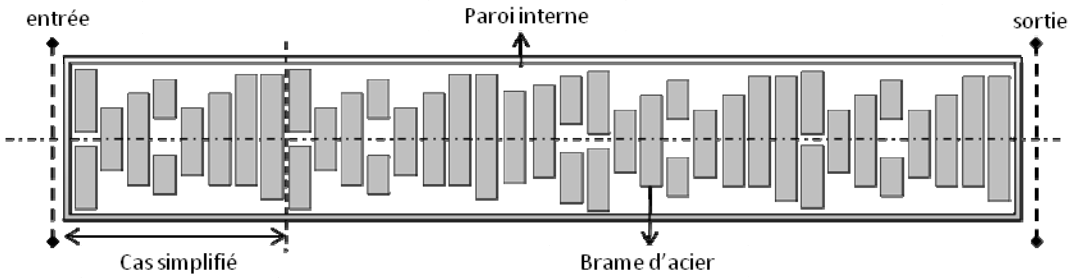


Figure 4 : Un plan de chargement des brames d'acier dans le four

Simulation dans MODRAY

MODRAY est utilisé dans cette étude comme référence pour la comparaison des résultats. C'est un outil de calcul des facteurs d'échanges radiatifs qui a été développé au Centre d'Efficacité Energétique des Systèmes (CES). Il est basé sur l'approximation des flux plans [3]. Il suffit de rappeler ici que MODRAY applique un maillage surfacique à l'enceinte et les objets qu'elle contient.

Pour les simulations, on considère un maillage surfacique de 10 × 2 × 2 mailles pour l'enceinte du four, 2 × 4 × 2 mailles pour les volumes de combustion, 4 × 2 × 2 mailles pour les brames d'acier, et 10 × 2 × 4 mailles pour les rails. Le volume libre du four est 'un milieu semi-transparent dont le coefficient d'absorption est supposé être égal à 0,1 m^{-1}. Les murs et les rails ont une émissivité de 0,85. Le coefficient d'absorption des volumes de combustion est de 0,6 et l'émissivité des brames d'acier est considérée comme étant égale à 0,9. Avec ce maillage, la simulation nécessite 55 secondes pour le cas simplifié et 800 secondes pour le four entier.

Simulation avec MACZM

La simulation du four entier et du cas simplifié (partie du four) sont désormais effectuées en utilisant MACZM avec la même configuration de l'enceinte et les mêmes propriétés radiatives. Un maillage rectangulaire uniforme est appliqué (voxels cubiques). Tout d'abord, une taille de voxels de 10 cm^3est considérée (figure 5.a). Ceci génère plus de neuf millions de voxels pour discrétiser la totalité du four. La simulation est trop lente pour ce maillage et seul le cas simplifié est considéré. Les simulations sont ensuite répétées avec des tailles de voxels de 20 cm^3, 40 cm^3, et 80 cm^3, générant 1,17 millions, 0,15 million et 18 000 voxels respectivement (figures 5.b à 5.d). Les résultats sont comparés aux résultats calculés par Modray en calculant l'erreur moyenne pondérée et sont présentés tableau 1. La précision de MODRAY a été validée expérimentalement dans des travaux précédents [3].

Tableau 1 - *Comparaison des résultats des différentes simulations du four de réchauffage dans MACZM*

Facteurs d'échange	**Temps de calcul sur CPU (s)**				**Erreur moyenne pondérée par rapport à MODRAY (%)**			
Taille de maille D (cm)	**10**	**20**	**40**	**80**	**10**	**20**	**40**	**80**
Parois – Brames	77.45	6.26	0.36	0.01	3.17	4.13	19.31	_
Parois – Supports	89.90	5.01	0.31	0.06	2.41	2.60	21.11	156.97
Parois – brames	8.19	0.72	0.03	0.00	5.18	11.95	29.73	_
Brames – Brames	2.42	0.22	0.05	0.00	1.24	19.44	47.05	_
Brûleurs – Supports	5.26	0.37	0.05	0.02	4.51	7.72	59.45	552.13
Parois – Parois	375	18.72	0.92	0.06	–	_	_	_
Brûleurs – Brûleurs	32459	527	5.90	0.08	4.28	4.83	6.29	13.49
Brûleurs – Brames	731	21.96	1.34	0.00	6.31	7.58	_	_
Brûleurs – Supports	1628	38.55	1.16	0.06	6.99	8.59	20.84	310.55
Parois – Brûleurs	6341	186	5.26	0.13	4.85	5.46	6.09	11.82
Temps de calcul total	**41 717 (s)**	**805 (s)**	**15.36(s)**	**0.42 (s)**				

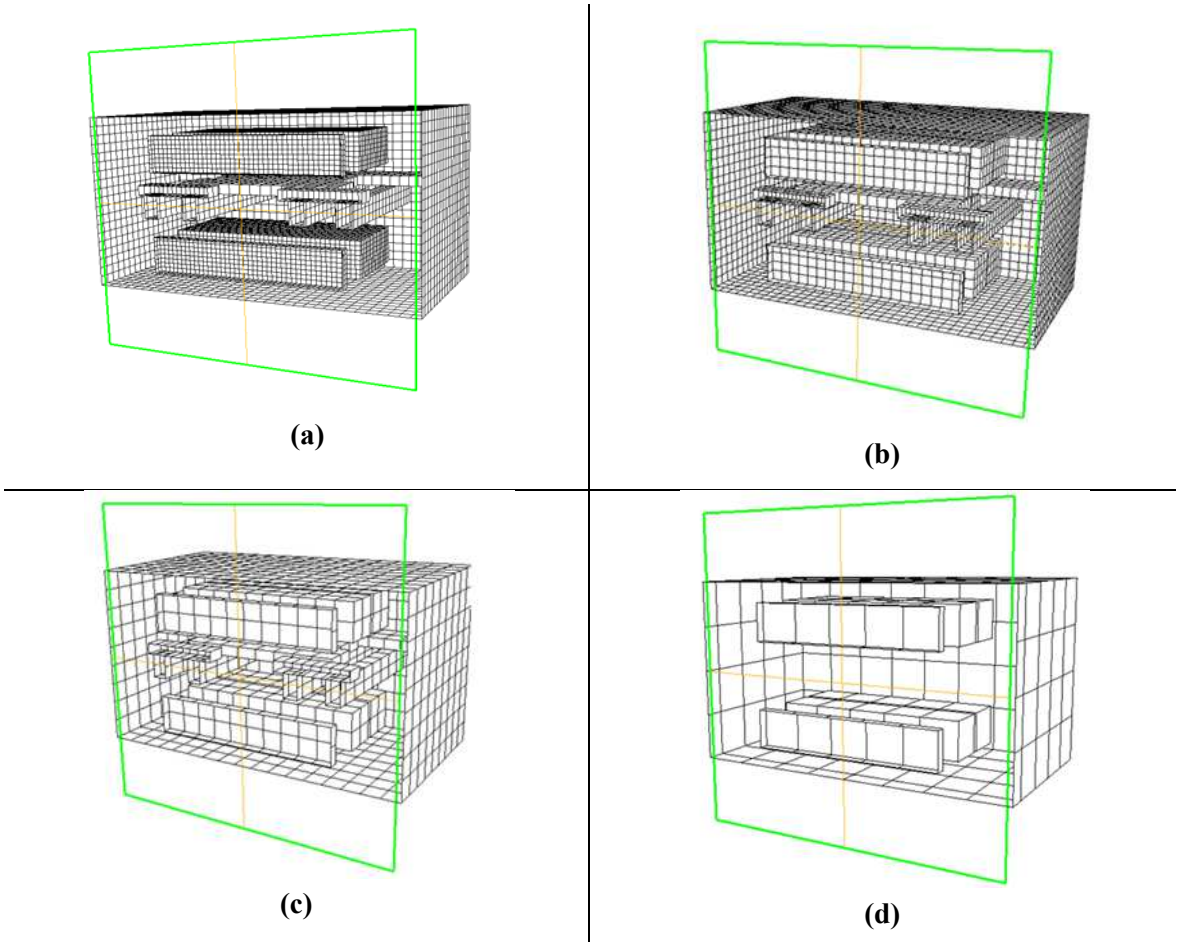


Figure 5 : Simulation du four de réchauffage dans MACZM. (a) Maillage de taille D = 10 cm (b) D = 20 cm (c) D = 40 cm (d) D = 80 cm

Erreur moyenne pondérée

L'erreur moyenne pondérée est calculée pour chaque catégorie de facteurs d'échange comme étant la moyenne pondérée des erreurs relatives correspondant à tous les facteurs d'échange de cette catégorie par rapport à MODRAY :

$$Erreur\ moyenne\ pondérée = \sum_i (e_i \times v_i) \Big/ \sum_i v_i \qquad (10)$$

où e_i est l'erreur relative pour un facteur d'échange i par rapport à sa valeur donnée par MODRAY et v_i sa valeur.

MACZM multi-grille

Considérons les résultats de MACZM avec le maillage le plus fin, de taille 10 cm^3. Le tableau 2 montre que tous les résultats sont précis. Cependant, lorsque la distance entre les objets est très grande les résultats sont moins précis. Ceci est dû au fait que les réseaux de neurones artificiels sont générés pour un nombre de voxels limité. Pour le maillage de dimensions 20 cm^3, l'erreur reste acceptable partout sauf pour les facteurs d'échange entre les brames elles-mêmes et entre les brames et les rails (tableau 2). Par conséquent, le maillage de dimensions 10 cm^3 est nécessaire

pour le calcul des facteurs d'échange entre brames et entre les brames et les rails. Les calculs des facteurs d'échange directs entre différents objets étant indépendants dans MACZM, il est donc possible d'adopter différents maillages pour le calcul des autres facteurs d'échange. En analysant plus en détail les résultats (cf. tableau 1), la configuration multi-grille peut être facilement déterminée en identifiant pour chaque catégorie de facteurs d'échange la plus grande dimension de voxel qui permet encore d'avoir une précision suffisante. En résultat, les voxels de 20 cm^3 sont les plus grands qui permettent d'obtenir des résultats corrects pour les facteurs d'échange entre les murs et les brames d'acier et entre les supports et les parois du four ainsi que entre les volumes de combustion et les supports. En revanche, les facteurs d'échange de chaleur entre les volumes de combustion eux-mêmes et entre les volumes des combustion et les parois du four ainsi que entre parois peuvent être effectués avec un maillage de 40 cm^3. Finalement, le maillage de 80 cm^3aboutit à des résultats erronés quel que soit le cas, parce qu'il il n'arrive plus à représenter correctement la géométrie du four, comme on peut le constater figure 5.d. Les résultats de l'approche multi-grille sont regroupés tableau 2.

Tableau 2 - M*aillage multi-grille appliqué au le four de réchauffage de brames*

Facteurs d'échange	**Taille de voxel (cm)**	**Temps CPU du cas simplifié (s)**	**Temps CPU du four entier (s)**	**Erreur moyenne pondérée (%)**
Parois – Brames	20	0.36	6.00	4.13
Parois – Supports	20	0.31	5.88	2.60
Parois – Brames	10	8.19	289.00	5.18
Brames – Brames	10	2.42	251.21	1.24
Brûleurs – Supports	20	0.37	10.56	7.72
Parois – Parois	40	0.92	134.32	_
Brûleurs – Brûleurs	40	5.90	86.57	6.29
Brûleurs – Brames	20	21.96	67.50	7.58
Brûleurs – Supports	20	38.55	403.40	8.59
Parois – Brûleurs	40	1.16	7.20	6.09
Temps de calcul total		**80.14 (s)**	**1 162 (s)**	

L'approche multi-grille permet d'obtenir des résultats précis tout en évitant toute perte de temps de calcul due à des maillages trop fins. En ajoutant les temps de calcul pour tous les facteurs d'échange, on trouve que la simulation du cas simplifié dans MACZM nécessite un temps CPU de 80 secondes alors que la simulation du four entier prend 1 162 secondes. Les temps de calcul pour la simulation du four entier sur CPU et sur GPU simultanément avec MACZM sont présentés tableau 3. Le rapport du temps CPU sur le temps GPU pour chaque catégorie du four entier dans MODRAY est donné dans la dernière colonne du tableau. Ce rapport varie entre 270 et 640. En effet, le temps de calcul GPU est influencé par la distance séparant les objets. Lorsque la distance

est plus longue, la ligne discrète est plus grande et le calcul du coefficient d'absorption moyen ralentit l'exécution sur GPU. Deplus, l'accélération par rapport au CPU est plus grande pour les facteurs d'échange volume-volume que pour les facteurs surface-surface. En moyenne, le temps de calcul sur GPU est 320 fois plus court que le temps CPU et permet de réduire le temps de simulation du four entier à quelques secondes. Ce temps de simulation rend la modélisation dynamique en temps réel du four possible et ouvre la possibilité au contrôle en ligne de haute précision du fonctionnement du four.

Tableau 3 - *Comparaison entre les temps de calcul CPU et GPU pour MACZM multi-grille appliquée au four de réchauffage de brames d'acier*

Facteurs d'échange	**Temps de calcul CPU (s)**	**Temps de calcul GPU (s)**	**Rapport des temps de calcul (CPU/GPU)**
Parois – Brames	6.00	0.019	315
Parois – Supports	5.88	0.018	326
Parois – Brames	289.00	0.996	290
Brames – Brames	251.21	0.931	270
Brûleurs – Supports	10.56	0.033	320
Parois – Parois	134.32	0.447	300
Brûleurs – Brûleurs	86.57	0.130	665
Brûleurs – Brames	67.50	0.153	421
Brûleurs – Supports	403.40	0.85	474
Parois – Brûleurs	7.20	0.016	450
Temps de calcul total	**1 162**	**3.59**	**322**

Références

1. W. W. Yuen, The Multiple Absorption Coefficient Zonal Method (MACZM), an Efficient Computational Approach for the Analysis of Radiative Heat Transfer in Multidimensional Inhomogeneous Nongray Media, *Numerical Heat Transfer*, Part B, vol. 49, pp. 89-103, 2006.
2. W. W. Yuen, Definition and Evaluation of Mean Beam Lengths for Applications in Multidimensional Radiatve Heat Transfer: A Mathematically Self-Consistent Approach, *J. Heat transfer*, vol.130, 2008.
3. B. Ghannam, M. Nemer, K. El Khoury, and W. Yuen, Experimental Validation of The Multiple Absorption Coefficient Zonal Method (MACZM) In a Dynamic Modeling of a Steel Rehating Furnace, *Numerical Heat Transfer*, Part A, vol. 58, pp. 1-19, 2010.
4. NVIDIA. CUDA Technology; http://www.nvidia.com/CUDA. 2007.
5. NVIDIA. CUDA Programming Guide http:// developer.download.nvidia.com/compute/DevZone/ docs/html/C/doc/CUDA_C_Programming_Guide.pdf. 2011.
6. Kaufaman and E. Shimony, 3D Scan-Conversion Algorithms for Voxel Based Graphics, proc. Workshop on Interactive 3D Graphics, ACM Press, New York, pp.45-75, 1986.

7. Ghannam, M. Nemer, K. El Khoury, and W. Yuen, An Efficient CPU-GPU Implementation of the Multiple Absorption Coefficient Zonal Method (MACZM), *Numerical Heat Transfer*, Part B, vol. 62, pp. 439-461, 2012.

B.3 Calcul des Facteurs de Transferts Radiatifs Totaux

B.3.1 Introduction

La majorité des méthodes numériques pour le calcul des échanges radiatifs sont basées sur le calcul des facteurs d'échanges directs (DEA) et les facteurs d'échanges totaux (TEA) [1-8]. Des temps de calcul trop longs sont généralement associés au calcul des TEA. Parmi les méthodes les plus performantes pour le calcul des TEA [9-13], l'algorithme des revêtements (PA) [14-17] est connu pour son efficacité et sa simplicité. Cet algorithme est basé sur un ensemble d'équations récursives. Des implémentations parallèles CPU et GPU efficaces pour le PA sont présentées. Néanmoins, la complexité du PA reste de l'ordre de N^3. Dans le but de réduire la complexité de l'algorithme et d'avoir une formulation plus flexible et plus parallélisable, une formulation non récursive du PA est effectuée dans ce travail : l'algorithme des revêtements non récursif (NRPA). Les équations du NRPA sont formulées par identification aux équations du PA donnant les TEA à partir des DEA. En outre, le NRPA s'écrit alors sous forme matricielle comme une opération de multiplication de matrices carrées. Ceci diminue la complexité de l'algorithme de l'ordre de $N^{2,38}$ par rapport au nombre de multiplications. L'implémentation du NRPA sur CPU et sur GPU est ensuite effectuée en se basant sur les librairies d'algèbre linéaire BLAS et CUBLAS respectivement. Enfin, une comparaison des performances des différentes implémentations est présentée montrant un temps d'exécution du NRPA jusqu'à 750 fois plus rapide que le PA initial.

B.3.2 Formule mathématique de l'algorithme des revêtements

L'algorithme des revêtements (PA) calcule les facteurs d'échanges totaux TEA à partir des facteurs d'échanges directs DEA en prenant en compte toutes les multi-réflexions que subit le rayonnement émis par une surface (ou volume) avant d'atteindre une autre surface (ou volume). Soit une enceinte contenant N facettes grises et diffuses en rayonnement (Figure 1). Dans le cas d'un milieu transparent, les facteurs d'échanges directs DEA entre les facettes sont liés aux facteurs de forme par :

$$\overline{S_i S_j} = A_i f_{ij} \tag{1}$$

$\overline{S_i S_j}$ étant le facteur d'échange direct entre la surface i et la surface j. Les facteurs d'échanges totaux (TEA) sont liés aux facteurs de transferts radiatifs de la même manière. L'application de l'algorithme des revêtements s'effectue en N étapes récursives. Les facettes de l'enceinte sont d'abord supposées noires en rayonnement. Ensuite, chaque étape consiste à faire correspondre à une facette S_k sa vraie emissivité et de recalculer tous les facteurs d'échange dans l'enceinte par les formules suivantes :

$$(\overline{S_k S_k})_k = \frac{A_k \varepsilon_k^2 (\overline{S_k S_k})_{k-1}}{A_k - \rho_k (\overline{S_k S_k})_{k-1}} \tag{2}$$

$$(\overline{S_i S_k})_k = \frac{A_k \varepsilon_k (\overline{S_i S_k})_{k-1}}{A_k - \rho_k (\overline{S_k S_k})_{k-1}} \qquad \begin{matrix} i = 0, 1, \dots N, \\ i \neq k \end{matrix} \tag{3}$$

$$\left(\overline{S_k S_j}\right)_k = \frac{A_k \varepsilon_k \left(\overline{S_k S_j}\right)_{k-1}}{A_k - \rho_k (\overline{S_k S_k})_{k-1}} \qquad \begin{matrix} j = 0, 1, \dots N, \\ j \neq k \end{matrix} \tag{4}$$

$$\left(\overline{S_i S_j}\right)_k = \left(\overline{S_i S_j}\right)_{k-1} + \frac{\rho_k (\overline{S_i S_k})_{k-1} \left(\overline{S_k S_j}\right)_{k-1}}{A_k - \rho_k (\overline{S_k S_k})_{k-1}} \qquad \begin{matrix} i, j = 0, 1, \dots N, \\ i, j \neq k \end{matrix} \tag{5}$$

où $\overline{S_i S_j}$ est le facteur d'échange avant le revêtement de la facette S_k. A_k est l'aire de S_k et $\overline{S_i \acute{S}_j}$ est le facteur d'échange après revêtement.

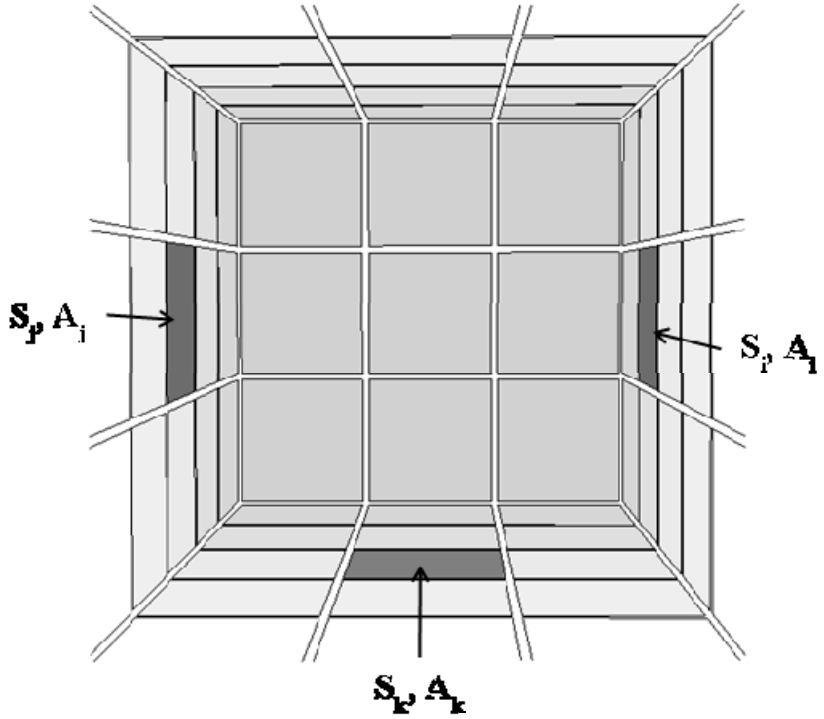


Figure 1: Enceinte avec N surface grises

Cai et El Khoury ont généralisé l'algorithme pour les enceintes délimitées par des facettes ayant des propriétés spéculaires [15]. D'autre part, l'extension de l'algorithme pour les volumes isotropiques en émission et en absorption se fait en remplacant la réflectivité par l'albedo de diffusion, et l'émissivité par 1 moins celui-ci, et A_k par $4k\nu_k$ [17]. Les équations des re-revêtements sont équivalentes aux équations (2) à (5) pour le cas des facettes grises et diffuses où ε_k est remplacé par $\varepsilon'_k/\varepsilon_k$ et ρ_k par $(\rho_k - \rho'_k)/\varepsilon_k^2$, ε'_k et ρ'_k étant les nouvelles émissivité et reflectivité de la facette S_k [15].

Finalement, la parallélisation du PA n'est pas faisable entre les différentes étapes de l'algorithme, parce qu'elles sont toutes dépendantes entre elles. En revanche, la parallélisation peut s'effectuer au sein de chaque étape vu que les calculs des facteurs d'échange dans une étape ne sont pas interdépendants et dépendent seulement des étapes précédentes.

B.3.3 L'algorithme des revêtements non-récursif (NRPA)

Afin de formuler l'expression de l'algorithme des revêtements non récursif, les variables suivantes sont introduites dans les équations (2) à (5)

$$\alpha_k = \frac{A_k \varepsilon_k^2}{A_k - \rho_k (\overline{S_k S_k})_{k-1}} \tag{6}$$

$$\beta_k = \frac{A_k \varepsilon_k}{A_k - \rho_k (\overline{S_k S_k})_{k-1}} \tag{7}$$

$$\gamma_k = \frac{\rho_k}{A_k - \rho_k (\overline{S_k S_k})_{k-1}} \tag{8}$$

Ensuite, un développement analytique des équations de l'algorithme des revêtements pour une enceinte avec trois facettes est effectué, permettant d'identifier une formule du NRPA qui est ensuite généralisé au cas avec N facettes par récurrence. La formulation peut être consultée dans 3.3, et la formule du NRPA résultante est la suivante :

$$\overline{S_i S_j} = \beta_i \beta_j \left[\overline{s_i s_j} + \sum_{t \in [1,n]}^{t \neq i,j} \left(\gamma_t \, \overline{s_i s_t} \, \overline{s_t s_j} \right) \right] \qquad i,j \in [1,N] \tag{9}$$

$$\overline{S_i S_i} = \alpha_i \left[\overline{s_i s_i} + \sum_{t \in [1,i-1]} \left(\gamma_t \, \overline{s_i s_t}^2 \right) \right] + \beta_i^2 \left[\sum_{t \in [i+1,n]} \left(\gamma_t \, \overline{s_i s_t}^2 \right) \right] \qquad i \in [1,N] \tag{10}$$

C'est la formulation du NRPA d'ordre 2. L'ordre 2 signifie que tous les éléments contenant la multiplication de plus de deux facteurs d'échange sont négligés dans la formulation. Les NRPA d'ordre supérieur sont formulés similairement. De ce fait une erreur est introduite. Pour un NRPA d'ordre m, l'erreur est comme suit :

$$erreur \; = O(\gamma^{m-1} \cdot \overline{s_i s_j}^m) \tag{11}$$

Pour le même ordre m, cette erreur augmente quand l'emissivité des surfaces diminue parce que la diminution de l'émissivité engendre une augmentation de γ. Plus évidemment pour un ordre supérieur, l'erreur est inférieure. D'autre part, il a été démontré par un exemple que la formule du NRPA permet de calculer tous les facteurs d'échange avec la même précision, indépendamment des différences de l'ordre de grandeur entre eux. Finalement, pour une émissivité des surfaces de l'ordre de 0,25, un NRPA d'ordre 6 permet de calculer les facteurs de transferts avec une erreur inférieure à 1 %.

B.3.4 Implémentation du PA et du NRPA et comparaisons des temps de calcul

Implémentation parallèle du PA sur CPU

La parallélisation sur CPU du NRPA est simple. Dans le cas d'un processeur à N cœurs, chaque cœur calcule 1 / p des surfaces d'échange, à chaque étape de revêtement. Il est à noter ici qu'à une étape k, il est nécessaire d'accéder aux surfaces d'échange $\overline{s_i s_j}$, $\overline{s_i s_k}$, $\overline{s_k s_j}$ et$\overline{s_k s_k}$ de l'étape $k-1$

afin de calculer la surface d'échange $\overline{S_i S_j}$ de l'étape k. Il est ensuite nécessaire que chaque cœur de processeur soit capable d'accéder à toutes les surfaces d'échange calculées à l'étape précédente. Ceci est garanti en insérant un point de synchronisation après chaque étape de revêtement. L'implémentation est effectuée sous OpenMP [20].

Implémentation parallèle du PA sur GPU

CUDA ("Compute Unified Device Architecture") est une extension du C, qui permet la programmation parallèle des GPU [18, 19]. Avec CUDA les mêmes instructions sont exécutées par plusieurs fils de calculs tournant en parallèle. En revanche, chaque GPU dispose de son propre jeu de mémoire. La mémoire DRAM est la mémoire principale du GPU. C'est une mémoire graphique à accès lent, appelée mémoire globale. Le temps d'accès à la mémoire DRAM est caché par la disponibilité d'une large bande passante et par l'exécution de milliers de fils parallèles en même temps. Ainsi, la parallélisation d'un programme sous CUDA nécessite des centaines et des milliers de fils qui exécutent les mêmes instructions en parallèle sur des données différentes. Il est aussi crucial de réduire et d'optimiser les accès en mémoire dans le but de cacher la faible latence de la mémoire DRAM. Ainsi, il serait possible de tirer le maximum de performance du GPU. En opposition, la particularité de l'algorithme des revêtements est la présence d'un nombre élevé d'accès en mémoire avec un très petit nombre d'opérations arithmétiques, comme il peut être observé dans les équations (2) à (5). En conséquence, le temps associé aux accès en mémoire doit être réduit. Ceci est fait en réarrangeant les surfaces d'échange dans une forme de matrice qui permet d'avoir des accès par blocs en mémoire.

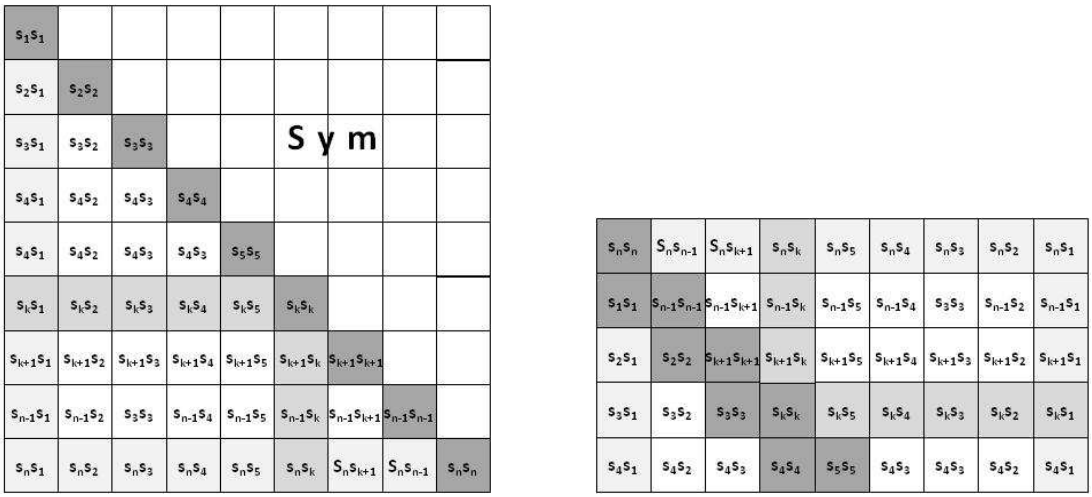


Figure 2 : Configuration GPU de la matrice des surfaces d'échange. (a) Matrice initiale; (b) Configuration optimale

Ceci s'effectue quand des fils parallèles voisins accèdent des emplacements mémoire voisins. Par ailleurs, afin de maximiser le nombre de fils parallèles, chaque fil s'occupe du calcul d'une seule surface d'échange à chaque étape de revêtement. La matrice des revêtements étant symétrique, elle n'est calculée qu'en moitié, ce qui réduit le nombre d'opérations mais en même temps est

inefficace sur GPU à cause de la dispersion des emplacements en mémoire. En conséquence, différentes possibilités de réarrangement des éléments de la matrice symétrique ont été testées, et la configuration présentée figure 3 a été retenue. Cette configuration permet d'avoir un temps de calcul très voisin de la moitié du temps de calcul de la matrice complète.

Implémentation CPU et GPU du NRPA

Les implémentations CPU et GPU du NRPA sont effectuées en se basant sur les librairies d'algèbres linéaires. Le NRPA est ramené à une forme matricielle qui consiste en une multiplication de deux matrices. La librairie d'algèbre linéaire (BLAS) est ensuite utilisée. De même, l'équivalent de BLAS pour CUDA, CUBLAS a été utilisé pour l'implémentation du NRPA sous CUDA.

Comparaison des temps de calcul

Les implémentations séquentielle et parallèle du PA du NRPA sont testées sur un CPU à quatre cœurs Intel Xeon E5507 2,26 GHz CPU et un GPU NVIDIA Tesla C 1060 avec 240 cœurs. Comme prévu, le PA s'exécute près de 4 fois plus rapidement sur quatre cœurs que sur un cœur du CPU (tableau 4). En revanche, le PA s'exécute 30 fois plus vite sur GPU par comparaison à l'exécution séquentielle sous CPU (tableau 1). On constate encore dans le tableau 1 que le rapport du temps de calcul GPU au temps de calcul CPU est plus élevé lorsque le nombre de surfaces est plus élevé. Ceci est dû à l'augmentation du nombre de fils parallèles sur le GPU, ce qui permet d'en tirer de meilleures performances. D'autre part, les temps de calcul pour la NRPA sont donnés tableau 2. En comparant ce tableau avec le tableau 1, on trouve que le temps de calcul du NRPA d'ordre 2 sur CPU est 45 fois plus rapide que le temps de calcul du PA séquentiel avec N = 1000, tout en augmentant jusqu'à 70 fois lorsque N atteint 10 000 surfaces.

Tableau 1 - Comparaison des temps de calcul du PA sequentiel et parallèle
(CPU : Intel Xeon E5507 CPU à 2.26GHz – GPU : NVIDIA Tesla C 1060)

Nombre de surfaces dans l'enceinte (N)	Temps CPU mono-coeur, en (s)	Temps CPU quatre-coeurs, en (s)	Temps GPU CUDA, en (s)
1000	1.29	0.35	0.073
2000	12.38	3.17	0.588
5000	331	86	13.1
10000	3202	810	98

L'augmentation du rapport du temps de calcul avec N est due au fait que la complexité du NRPA en fonction de N est inférieure à celle du PA. De même l'exécution du NRPA sur GPU se fait de 25 à 35 fois plus rapidement que pour le PA. Enfin, le temps de calcul GPU du NRPA d'ordre 2 est 400 à 750 fois plus rapide que celui du PA exécuté en séquentiel sur CPU. Enfin, les temps d'exécution du NRPA d'ordre 4 et 6 sont respectivement deux et trois fois plus longs que les temps de calcul du NRPA d'ordre 2.

Tableau 2 – Comparaison des temps de calcul du NRPA sequentiel et parallèle
(CPU : Intel Xeon E5507 CPU à 2.26GHz – GPU : NVIDIA Tesla C 1060)

Nombre de surfaces dans l'enceinte (N)	Temps CPU mono-Cœur, en (s)			Temps GPU CUDA, en (s)		
	Order 2	Order 4	Order 6	Order 2	Order 4	Order 6
1000	0.029	0.068	0.11	0.0032	0.0069	0.01
2000	0.3	0.59	0.87	0.028	0.055	0.08
5000	5.736	10.2	15.1	0.48	0.87	1.29
10000	45	71	97	4.29	7.4	10.6

Références

1. Becker, E. B., G. F. Carey, and J. T. Oden, Finite Elements An Introduction, Prentice Hall, Englewood Cliffs, NJ, vol. 1, 1981.
2. Raaque, M. M, D. E. Klein, and J. R. Howell, Finite Elements Solution of Radiative Heat Transfer in a Two-Dimensional Rectangular Enclosure with Gray Participating Media, J. Heat Transfer, vol. 105, no.4, pp. 933-934, 1983.
3. Hottel, H. C., and E.S. Cohen, Radiant Heat Exchange in a Gas-Filled Enclosure: Allowance for Nonuniformity of Gas Tempereature, AIChE J., Vol. 4, no.1, pp 3-14, 1983.
4. Noble, James J., The Zone Method: Explicit Matrix Relations for Total Exchange Areas, int. J. Heat and Mass Tranfer, vol. 18, no. 2, pp 1153-1169, 1965.
5. Yuen, W. W., and E. E. Takara, Development of a Generalized Zonal Method for the Analysis of Radiative Heat Transfer in Absorbing and Anisotropically Scattering Media, in K.-Vafai and J. L. S. Chen (eds.), Numerical heat Transfer, ASME HTD-vol.130, pp 123-132, 1990.
6. W. W. Yuen, The Multiple Absorption Coefficient Zonal Method (MACZM), an Efficient Computational Approach for the Analysis of Radiative Heat Transfer in Multidimensional Inhomogeneous Nongray Media, Numerical Heat Transfer, Part B, vol. 49, pp. 89-103, 2006.
7. W. W. Yuen, Definition and Evaluation of Mean Beam Lengths for Applications in Multidimensional Radiatve Heat Transfer: A Mathematically Self-Consistent Approach, J. Heat Transfer, vol. 130(11), 114507 (5 pages), 2008.
8. B. Ghannam, M. Nemer, K. El Khoury, and W. Yuen, Experimental Validation of The Multiple Absorption Coefficient Zonal Method (MACZM) In A Dynamic Modeling Of A Steel Rehating Furnace, Numerical Heat Transfer, Part A, vol. 58, pp. 1-19, 2010.
9. Hottel, Hoyt, C., Radiant-Heat Transmission, in William H. McAdams (ed.), Heat Transmission, 3d ed., chap. 4, McGraw-Hill, New York, 1954.
10. Poljak. G., Analysis of Heat Interchange by radiation between Diffuse Surfaces, tch. Phys. USSR, vol. 1, nos. 5and 6, pp. 555-590, 1935.
11. Y. Y. Jiang, A Two-Step Strategy for Numerical Simulation of Radiative Transfer with Anisotropic Scattering and Reflection, J. Quant. Spectroscop. Radiat. Transfer, vol. 109, pp. 636-649, 2008.
12. Gebhart, B., Heat Transfer, 2d ed., pp. 150-163, McGraw-Hill, New York, 1971.

13. Gebhart, B., Surface temperature calculations in radiant Surroundings of Arbitrary Complexity fro Gray, Diffuse Radiation, Int. J. Heat mass Transfer, vol. 3 no. 4, pp. 341-346, 1961.

14. D. K. Edwards, The Plating Algorithm for Radiation Script-F Transfer Factor, ASME J. Heat Transfer, vol. 108, pp. 237-238, 1986.

15. G. El Hitti, M. Nemer, and K. El Khoury, Reducing CPU Time for Radiative Exchange and Transient Heat Transfer Analysis Using Zone Method, Numerical Heat Transfer, Part B, vol. 56, pp. 23-37, 2009.

16. G. El Hitti, M. Nemer, K. El Khoury, and D. Clodic, "The re-plating algorithm for radiation total exchange area calculation", Numerical Heat Transfer, Part B, vol. 57, pp. 110-125, 2010.

17. V. Strassen., Gaussian Elimination is not Optimal, Numer. Math., vol. 13, pp. 354-356, 1969.

18. NVIDIA, CUDA Technology; http://www.nvidia.com/CUDA, 2007.

19. David B. Kirk and Wen-mei W. Hwu, Programming Massively Parallel Processors: A Hands-on Approach (Applications of GPU Computing Series), Elsevier Inc. 2010.

20. The OpenMP website, www.openmp.org.

B.4 Modélisation de la Diffusion 3D par la Méthode des Différences Finies

B.4.1 Introduction

De nombreuses applications scientifiques et d'ingénierie sont basées sur la solution d'équations différentielles partielles. Dans ce travail, on présente une solution efficace de l'équation différentielle partielle de la diffusion de la chaleur en 3D. Cette solution est basée sur une optimisation entre les méthodes de calcul en différences finies, des solveurs efficaces et des schémas de parallélisation efficaces sous CUDA pour application au calcul sur GPU.

Un rappel de la discrétisation en différences finies est d'abord présenté [1,2]. Les schémas de résolution directs résultant de la discrétisation en différences finies sont les schémas explicite, implicite et le schéma de Crank-Nicolson [3]. Le schéma explicite possède une condition de stabilité tandis que le schéma implicite et le schéma de Crank-Nicolson sont inconditionnellement stables. De plus, le schéma de Crank-Nicolson est le plus précis avec une erreur de troncature du second ordre par rapport au temps et à l'espace. Néanmoins, ces deux derniers schémas font appel à la résolution d'une matrice de taille $N^3 \times N^3$ pour un maillage ayant N points dans chaque dimension, ce qui devient rapidement très coûteux en temps de calcul pour un grand nombre de mailles. La solution à ce problème réside dans l'application des méthodes qui divisent le calcul en trois étapes à directions alternées, notamment la méthode localement unidimensionnelle (LOD) [17, 18]. Cette méthode permet la résolution d'une grille avec N points dans chaque dimension en effectuant la solution de N^2 systèmes tridiagonaux à N équations, ce qui est nettement moins complexe en nombre d'opérations que la résolution d'une matrice de taille $N^3 \times N^3$ nécessaire avec le schéma de Crank-Nicolson.

D'autre part, le meilleur algorithme pour la résolution des systèmes tridiagonaux d'équations linéaires sur CPU est l'algorithme de Thomas [1]. Cependant, cet algorithme est complètement séquentiel. En revanche les algorithmes de réduction cyclique (CR) [11-13] et de réduction cyclique parallèle (PCR) [20] permettent la résolution des systèmes tridiagonaux en parallèle. Ces algorithmes sont présentés et comparés dans le corps du chapitre.

Enfin, deux schémas de parallélisation efficace sous CUDA [9, 10] sont présentés. Le premier schéma permet d'avoir un nombre maximal de fils s'exécutant en parallèle et le deuxième schéma met en jeu un nombre de fils parallèles plus petit permettant la résolution avec l'algorithme de Thomas. Les deux schémas de parallélisation sont ensuite testés avec les différents solveurs et comparés. Le 1er schéma ave PCR est le plus efficace pour des grilles avec des petits nombres de mailles et le 2ème schéma appliqué avec l'algorithme de Thomas dépasse ce dernier dès qu'un certain nombre de mailles est atteint. Finalement, un solveur hybride est conçu à partir des deux schémas afin de tirer le maximum de performances. Le schéma hybride présente la propriété de garantir des accélérations de quelques centaines de fois par rapport au temps CPU, même pour des grilles à petit nombre de mailles.

B.4.2 Formulation de l'équation de la diffusion de chaleur 3D selon les méthodes des différences finies

Equation de la chaleur tridimensionnelle

Soit la diffusion de chaleur linéaire en 3D dans une brame parallélipipédique de dimensions (L_x, L_y, L_z) et dont les propriétés physiques sont isotropiques. En considérant un système de coordonnées (O, x, y, z), l'équation de la diffusion de chaleur 3D dans la brame s'écrit :

$$\frac{\partial T}{\partial t} = \alpha\left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + \frac{\partial^2 T}{\partial z^2}\right) \tag{1}$$

où $T = T(x, y, z, t)$ et $\alpha = k/\rho c_p$. k étant la conductivité thermique, ρ la densité et c_p la capacité thermique, à la position (x, y, z) et au temps t.

Discrétisation de l'équation différentielle partielle (EDP) de la diffusion de la chaleur en 3D

La formulation discrète de l'EDP consiste à définir la température sur un nombre fini de points dans l'espace et dans le temps. Pour ce, l'espace continu (L_x, L_y, L_z) est discrétisé en un nombre fini de points (I, J, K), séparés par les distances $(\Delta x, \Delta y, \Delta z)$. L'espace de temps est aussi discrétisé en un nombre N de points séparés par un intervalle de temps constant Δt. La température au point (i, j, k) et au pas de temps n est ensuite notée $T_{i,j,k}^n$:

$$T_{i,j,k}^n = T(i\Delta x, j\Delta y, k\Delta z, n\Delta t) \tag{2}$$

L'approximation en différences finies de l'équation (1) est obtenue en développant les séries de Taylor pour la température [1]. L'approximation à l'ordre 1 de la dérivée de la température par rapport au temps est donnée par :

$$\frac{\partial T_{i,j,k}^n}{\partial t} = \frac{T_{i,j,k}^{n+1} - T_{i,j,k}^n}{\Delta t} + O(\Delta t) \qquad \textit{(avancé)}$$

$$\frac{\partial T_{i,j,k}^n}{\partial t} = \frac{T_{i,j,k}^n - T_{i,j,k}^{n-1}}{\Delta t} + O(\Delta t) \qquad \textit{(retardé)} \tag{3}$$

$$\frac{\partial T_{i,j,k}^n}{\partial t} = \frac{T_{i,j,k}^{n+1} - T_{i,j,k}^{n-1}}{2\Delta t} + O(\Delta t^2) \qquad \textit{(centrale)}$$

L'approximation de la dérivée seconde de la température par rapport à une variable d'espace est donnée par la formulation centrale au second ordre des différences finies :

$$\frac{\partial^2 T_{i,j,k}^n}{\partial x^2} = \frac{T_{i+1,j,k}^n - 2T_{i,j,k}^n + T_{i-1,j,k}^n}{(\Delta x)^2} + O(\Delta x^2) = \frac{\partial_x^2 T^n}{(\Delta x)^2} + O(\Delta x^2) \tag{4}$$

La solution numérique de l'équation discrétisée possède des critères de consistance, de convergence et de stabilité. Dans ce qui suit, différents schémas de discrétisation sont considérés. Tous les schémas considérés ont les propriétés de consistence et convergence. Seul le critère de stabilité varie.

Schémas de résolution

Selon que la mise en équation considère la discrétisation avancée (Equation (1)) ou la discrétisation retardée (Equation (3)) par rapport au temps, le schéma de discrétisation est respectivement explicite ou implicite [1, 2]. Le schéma explicite est stable si et seulement si $\left(\frac{\alpha\Delta t}{(\Delta x)^2}+\frac{\alpha\Delta t}{(\Delta y)^2}+\frac{\alpha\Delta t}{(\Delta z)^2}\right)\leq\frac{1}{2}$, tandis que le schéma implicite est inconditionnellement stable. Dans les deux cas, l'erreur de troncature est du premier ordre par rapport au temps, $O(\Delta t,\Delta x^2,\Delta y^2,\Delta z^2)$.

L'erreur de troncature est améliorée en considérant la moyenne arithmétique de la dérivée de la température par rapport aux variables d'espace des deux schémas explicite et implicite. Le schéma de discrétisation obtenu est le schéma de Crank-Nicolson :

$$\frac{T_{i,j,k}^{n+1}-T_{i,j,k}^{n}}{\Delta t}=\frac{\alpha}{2}\left[\frac{\partial_x^2T^n+\partial_x^2T^{n+1}}{(\Delta x)^2}+\frac{\partial_y^2T^n+\partial_y^2T^{n+1}}{(\Delta y)^2}+\frac{\partial_z^2T^n+\partial_z^2T^{n+1}}{(\Delta z)^2}\right] \tag{5}$$

où $i\in[1,I]$, $j\in[1,J]$ et $k\in[1,K]$.

On rappelle que le schéma de Crank-Nicolson est inconditionnellement stable [1] et il est démontré dans [16, 17] que l'erreur de troncature est de l'ordre $O(\Delta t^2,\Delta x^2,\Delta y^2,\Delta z^2)$.

En réarrangeant les termes de l'équation (5), on aura :

$$T_{i,j,k}^{n+1}-\frac{r_x}{2}\partial_x^2T^{n+1}-\frac{r_y}{2}\partial_y^2T^{n+1}-\frac{r_z}{2}\partial_z^2T^{n+1}=T_{i,j,k}^{n}+\frac{r_x}{2}\partial_x^2T^{n}+\frac{r_y}{2}\partial_y^2T^{n}+\frac{r_z}{2}\partial_z^2T^{n} \tag{6}$$

où $i\in[1,I]$, $j\in[1,J]$ et $k\in[1,K]$.

La résolution du système obtenu par l'équation (6) revient à résoudre une matrice carrée de taille $(I\times J\times K)^2$. Autrement dit, si $I=J=K=N$, la résolution du problème revient à résoudre une matrice de taille $N^3\times N^3$. Notons que la mise en équation du schéma implicite aboutit à une matrice de mêmes dimensions.

Méthode unidimensionnelle localement (LOD)

La méthode LOD a été présentée par D'yakonov [17] et Ianenko [18]. La méthode consiste à diviser le pas de temps en trois étapes successives. Dans chacune des trois étapes, l'équation de la diffusion est résolue selon une seule dimension de l'espace pour chaque couple de variables des deux autres dimensions et en se basant sur le schéma de Crank-Nicolson. Ceci revient à résoudre le système d'équations suivant :

step 1: for $i\in[1,I]$ $$\frac{\mathrm{T}_{\mathrm{i,j,k}}^{\mathrm{n}^*}-\mathrm{T}_{\mathrm{i,j,k}}^{\mathrm{n}}}{\Delta\mathrm{t}}=\frac{\alpha}{2}\left[\frac{\partial_{\mathrm{x}}^2\mathrm{T}^{\mathrm{n}}+\partial_{\mathrm{x}}^2\mathrm{T}^{\mathrm{n}^*}}{(\Delta\mathrm{x})^2}\right] \quad j\in[1,J] \text{ and } k\in[1,K] \tag{7.a}$$

step 2: for $j\in[1,J]$ $$\frac{\mathrm{T}_{\mathrm{i,j,k}}^{\mathrm{n}^{**}}-\mathrm{T}_{\mathrm{i,j,k}}^{\mathrm{n}^*}}{\Delta\mathrm{t}}=\frac{\alpha}{2}\left[\frac{\partial_{\mathrm{y}}^2\mathrm{T}^{\mathrm{n}^*}+\partial_{\mathrm{y}}^2\mathrm{T}^{\mathrm{n}^{**}}}{(\Delta\mathrm{y})^2}\right] \quad i\in[1,I] \text{ and } k\in[1,K] \tag{7.b}$$

step 3: for $k\in[1,K]$ $$\frac{\mathrm{T}_{\mathrm{i,j,k}}^{\mathrm{n}+1}-\mathrm{T}_{\mathrm{i,j,k}}^{\mathrm{n}^{**}}}{\Delta\mathrm{t}}=\frac{\alpha}{2}\left[\frac{\partial_{\mathrm{z}}^2\mathrm{T}^{\mathrm{n}^{**}}+\partial_{\mathrm{z}}^2\mathrm{T}^{\mathrm{n}+1}}{(\Delta\mathrm{z})^2}\right] \quad i\in[1,I] \text{ and } j\in[1,J] \tag{7.c}$$

La LOD est inconditionnellement stable et possède une erreur de troncation de l'ordre $O(\Delta t^2, \Delta x^2, \Delta y^2, \Delta z^2)$.

En développant les dérivées et en réarrangeant les termes de l'Equation (7.a), on obtient:

$$-r_x \mathrm{T}_{\mathrm{i-1,j,k}}^{\mathrm{n}^*} + 2(1+r_x)\mathrm{T}_{\mathrm{i,j,k}}^{\mathrm{n}^*} - r_x \mathrm{T}_{\mathrm{i+1,j,k}}^{\mathrm{n}^*} = r_x \mathrm{T}_{\mathrm{i-1,j,k}}^{\mathrm{n}} + 2(1-r_x)\mathrm{T}_{\mathrm{i,j,k}}^{\mathrm{n}} + r_x \mathrm{T}_{\mathrm{i+1,j,k}}^{\mathrm{n}} \quad (8)$$

pour $j \in [0, J]$ et $k \in [0, k]$.

De même, pour les équations (7.b) et (7.c).

La résolution de l'équation (8) revient donc à résoudre $J \times K$ systèmes d'équations tridiagonaux :

$$\begin{aligned}
b_1 x_1 + c_1 x_2 &= u_1 \\
a_2 x_1 + b_2 x_2 + c_2 x_3 &= u_2 \\
a_3 x_2 + b_3 x_3 + c_3 x_4 &= u_3 \\
\ddots &\ \vdots \\
a_{n-1} x_{n-2} + b_{n-1} x_{n-1} + c_{n-1} x_n &= u_{n-1} \\
b_{n-1} x_{n-1} + c_{n-1} x_n &= u_n
\end{aligned} \quad (9)$$

où $j \in [1, J]$ et $k \in [1, K]$ et :

$$a_i = -r_x \quad (10.\mathrm{a})$$

$$b_i = 2(1+r_x) \quad (10.\mathrm{b})$$

$$c_i = -r_x \quad (10.\mathrm{c})$$

$$x_i = \mathrm{T}_{\mathrm{i,j,k}}^{\mathrm{n}^*} \quad (10.\mathrm{d})$$

$$u_i = r_x \mathrm{T}_{\mathrm{i-1,j,k}}^{\mathrm{n}} + 2(1-r_x)\mathrm{T}_{\mathrm{i,j,k}}^{\mathrm{n}} + r_x \mathrm{T}_{\mathrm{i+1,j,k}}^{\mathrm{n}} \quad (10.\mathrm{e})$$

Enfin, si $I = J = K = N$, la solution de la PDE de la diffusion de la chaleur par la méthode LOD se réduit à la résolution de $3N^2$ matrices tri-diagonales de taille $N \times N$. La résolution de ce système est beaucoup moins complexe que la résolution du système qui résulte de la méthode de Crank-Nicolson.

Notons que d'autres méthodes ayant une complexité de résolution similaire ont été présentées dans la version en anglais. Ces méthodes sont les méthodes implicites à directions alternées ADI de Peaceman-Rachford [4], Brian [5] et de Douglass-Gunn [6,7], et la méthode explicite à directions alternées [8]. Néanmoins, il a été démontré que la LOD aboutit aux schémas de résolution les plus efficaces et présente la meilleure précision.

B.4.3 Solveurs efficaces de systèmes tridiagonaux d'équations linéaires

La résolution de la diffusion par la LOD revient à résoudre plusieurs systèmes tridiagonaux d'équations linéaires (Equation (9)) qui s'écrivent sous la forme matricielle suivante :

$$\begin{bmatrix} b_1 & c_1 & 0 & 0 & 0 & 0 \\ a_2 & b_2 & c_2 & 0 & 0 & 0 \\ 0 & a_3 & b_3 & c_3 & 0 & 0 \\ 0 & 0 & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & a_{n-1} & b_{n-1} & c_{n-1} \\ 0 & 0 & 0 & 0 & a_n & b_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_{n-1} \\ u_n \end{bmatrix} \tag{11}$$

Algorithme de Thomas

L'algorithme de Thomas est l'algorithme classique pour la résolution des systèmes tridiagonaux d'équations linéaires [27]. Cet algorithme est l'application de l'élimination de Gauss au cas particulier des systèmes tridiagonaux. L'algorithme de Thomas est divisé en deux étapes. La première étape consiste à transformer la matrice en une matrice diagonale supérieure par la phase d'élimination suivante :

$$c_1' = \frac{c_1}{b_1} \tag{12.a}$$

$$u_1' = \frac{c_1}{b_1} \tag{12.b}$$

Et pour $i = 2, \ldots, n$:

$$c_i' = \frac{c_i}{b_i - c_{i-1}' a_i} \tag{13.a}$$

$$u_i' = \frac{u_i - u_{i-1}' a_i}{b_i - c_{i-1}' a_i} \tag{13.b}$$

La deuxième étape consiste à calculer les variables par substitution inverse :

$$x_n = u_n' \tag{14}$$

Et pour $i = n - 1, \ldots, 1$:

$$x_i = u_i - c_i' x_{i+1} \tag{15}$$

D'après ce qui précède, l'algorithme de Thomas est séquentiel et demande $2n$ étapes et $8n$ opérations.

Réduction cyclique (CR)

L'algorithme de réduction cyclique consiste en deux phases, la première phase est une phase de réduction et la seconde est une phase de substitution inverse (Figure 1).

Chaque étape de la réduction réduit le système d'équation en moitié en éliminant toutes les équations impaires. Soient les équations $i - 1$, i et $i + 1$:

$$a_{i-1}x_{i-2} + b_{i-1}x_{i-1} + c_{i-1}x_i = u_{i-1} \tag{16.a}$$

$$a_i x_{i-1} + b_i x_i + c_i x_{i+1} = u_i \tag{16.b}$$

$$a_{i+1}x_i + b_{i+1}x_{i+1} + c_{i+1}x_{i+2} = u_{i+1} \tag{16.c}$$

Afin d'éliminer x_{i-1} et x_{i+1}, l'équation (16.a) est multipliée par $A_i = -a_i/b_{i-1}$ et l'équation (16.c) est multipliée par $C_i = -c_i/b_{i+1}$. Les trois équations sont ensuite additionnées, pour donner :

$$a'_i x_{i-2} + b'_i x_i + c'_i x_{i+2} = u'_i \tag{17}$$

Où a'_i, b'_i et c'_i sont les nouveaux coefficients pour l'équation i et sont donnés par :

$$a'_i = A_i a_{i-1} \tag{18.a}$$

$$b'_i = b_i + A_i c_{i-1} + C_i a_{i+1} \tag{18.b}$$

$$c'_i = C_i c_{i+1} \tag{18.c}$$

$$u'_i = u_i + A_i u_{i-1} + C_i u_{i+1} \tag{18.d}$$

La phase de réduction se termine quand il ne reste qu'une seule équation ou un système de deux équations à deux inconnues. L'équation oule système à deux équations est ensuite résolu afin de commencer la substitution inverse.

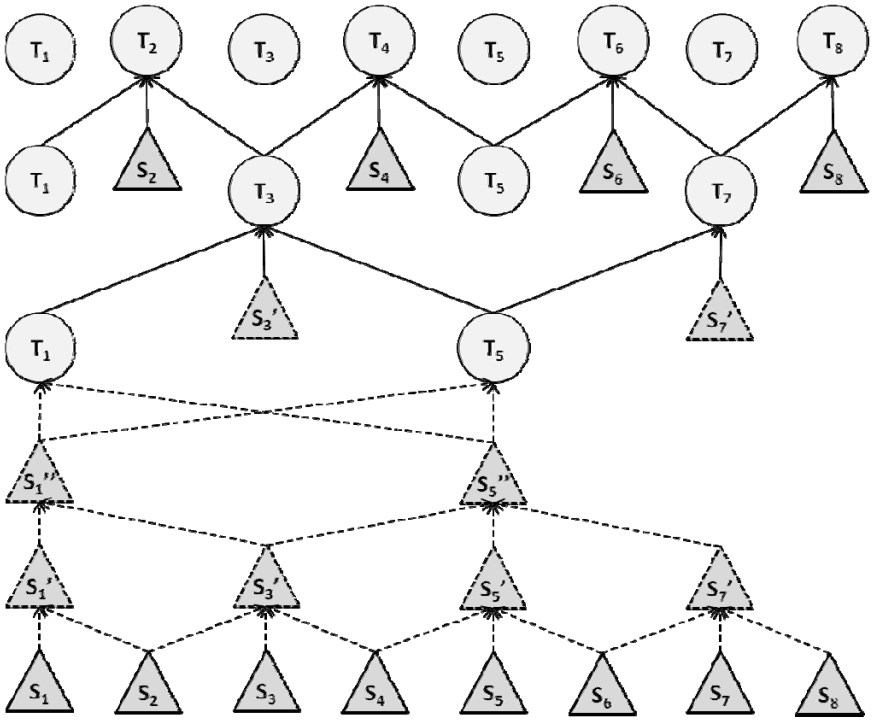


Figure 1 : Algorithme de réduction cyclique (CR)

Chaque étape de substitution inverse calcule la valeur des variables impaires en substituant les variables impaires par leurs valeurs déjà calculées:

$$x_i = \frac{u'_i - a'_i x_{i-1} - c'_i x_{i+1}}{b'_i} \tag{38}$$

La substitution se termine quand toutes les variables sont connues.

Enfin, la réduction cyclique met en jeu 17n opérations et $2\log_2 n - 1$ étapes séquentielles, chacune de ces étapes étant parallélisable.

Réduction cyclique parallèle (PCR)

L'algorithme de réduction cyclique parallèle [18] est une variante de la réduction cyclique. Il permet d'avoir plus de parallélisation sur l'ensemble des étapes de l'algorithme. La PCR consiste en une seule phase de réduction. A chaque étape de réduction, chacun des systèmes d'équations est coupé en deux systèmes au lieu d'être réduit à un système de demi-taille comme dans CR. Les deux nouveaux systèmes correspondent aux équations paires ou impaires (figure 2). Toutes les variables sont ainsi connues à la fin de la phase de réduction.

Finalement, l'algorithme de réduction cyclique parallèle s'exécute en $12n\log_2 n$ et $\log_2 n$ étapes, chacune des étapes étant éventuellement parallélisable. Comparé à CR, PCR permet d'avoir plus de parallélisation, mais aussi plus d'opérations algébriques asymptotiquement.

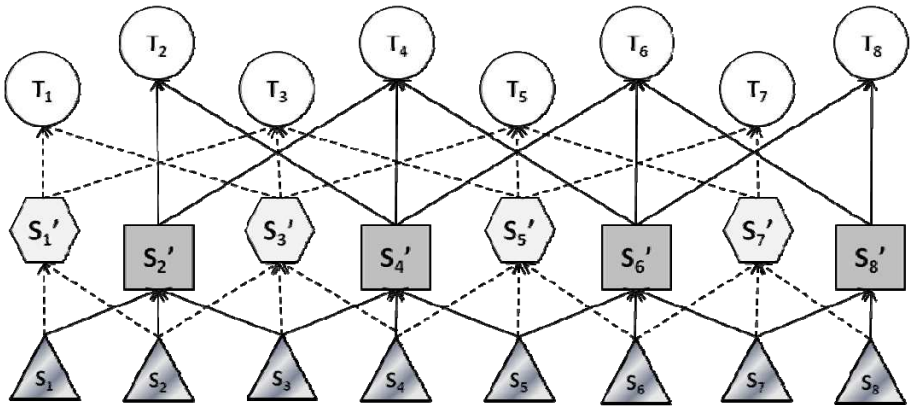


Figure 2 : Algorithme de reduction cyclique parallèle

B.4.4 Parallélisation de la LOD et implémentation dans CUDA

Chaque étape de la LOD consiste à calculer les termes de N^2 matrices tridiagonales de taille $N \times N$ et de les résoudre. La mise en équation des matrices ne dépend que des températures de l'étape de temps précédente, ce qui rend toutes les matrices indépendantes entre elles et par suite exécutables en parallèle. La parallélisation s'effectue ensuite pour chacune des étapes de la LOD (Equations 7.a à 7.c). Deux schémas de parallélisation efficaces sont considérés et sont présentés ci-après. Afin de simplifier la présentation, le nombre de mailles est supposé égal dans les trois direction x, y et z.

Parallélisation maximale 3D

Cette méthode permet d'avoir un nombre maximal de fils s'exécutant en parallèle. La parallélisation consiste à mettre en équation et résoudre chaque matrice tridiagonale par un bloc de N fils. Chaque bloc aura donc N équations à établir (Equations 7.a à 7.c). En conséquence, un bloc est formé de N fils, dont chaque fil s'occupe d'établir une seule équation. Il y aura donc N^2 blocs dont chaque bloc contient N fils, soit au total N^3 fils s'exécutant en parallèle.

En outre, d'après l'équation 7, on remarque que le calcul des coefficients de chaque équation nécessite la lecture de trois températures et donc trois accès en mémoire. L'accès en mémoire sur GPU étant lent, les températures de chaque bloc sont lues collectivement par les N fils du bloc et placées en mémoire partagée. Cela aboutit à la lecture d'une température par fil et par suite à un nombre minimal d'accès en mémoire.

Finalement, chaque bloc résout le système d'équations tridiagonal établi par l'algorithme CR ou PCR. L'algorithme de Thomas n'est pas applicable ici, du fait qu'il n'est pas parallélisable.

Parallélisation par plan

Dans cette méthode, chaque matrice tridiagonale est mise en equation par un seul fil de calcul. Les accès en mémoire sont optimisés, en effectuant des accès en mémoire voisins par les fils de calcul voisins, ce qui aboutit à des accès en mémoire par blocs. Cette méthode de parallélisation met en oeuvre N^2 fils parallèles et permet la résolution des systèmes tridiagonaux par l'algorithme de Thomas. La résolution par les algorithmes CR et PCR étant toujours possible et faisant appel à N^3 fils parallèles pour la phase de résolution.

Analyse des performances et comparaisons

L'implémentation des différentes méthodes est effectuée sous C et CUDA. Les performances sont testées sur un ordinateur équipé d'un CPU Intel Xéon E5507 à 2,26 GHz et d'une carte graphique « GPU » NVIDIA Tesla GPU C 1060.

Les figures 3.a - 3.e montrent les performances des différentes implémentations séquentielles de la LOD sur CPU et des différentes implémentations parallèles sur GPU pour un nombre de mailles allant de $16 \times 16 \times 16$ jusqu'à $256 \times 256 \times 256$. LOD1 correspond au premier schéma de parallélisation, la parallélisation maximale 3D et LOD2 correspond au deuxième schéma de parallélisation, la parallélisation en plans. Comme on peut observer sur la figure 3, avec le même schéma de parallélisation, les performances de CR sont toujours inférieures au PCR. En effet, même si CR met en jeu un nombre d'opérations plus petit pour certains nombres de mailles, le nombre d'étapes est toujours deux fois plus grand et son taux de parallélisation est beaucoup plus faible, ce qui rend le PCR plus rapide quelle que soit la taille de la grille. Un solveur hybride CR et PCR a été développé par Zhang et al. [10], qui permet de profiter de la parallélisation de PCR sur les étapes du CR où la parallélisation est faible. Néanmoins cet algorithme n'est que légèrement plus puissant que PCR et sur des nombres de mailles très grands.

D'autre part, on trouve que l'algorithme de Thomas, avec le schéma de parallélisation par plans, permet d'obtenir la plus grande performance à partir d'un certain nombre de mailles. En effet, à partir d'un certain nombre de mailles, la parallélisation devient suffisante pour cet algorithme afin de tirer profit des performances du GPU tout en ayant un nombre d'opérations plus petit que pour CR et PCR. Une solution hybride entre les deux implémentations les plus performantes, LOD1 avec PCR et LOD2 avec Thomas est analysée dans le paragraphe suivant.

Il a été observé que l'exécution sur GPU assure des temps de calculs plus rapides que les temps de calculs sur CPU, allant d'un facteur 10 pour les maillages les plus gros $(16 \times 16 \times 16)$ jusqu'à 250 pour les maillages les plus fins $(256 \times 256 \times 256)$.

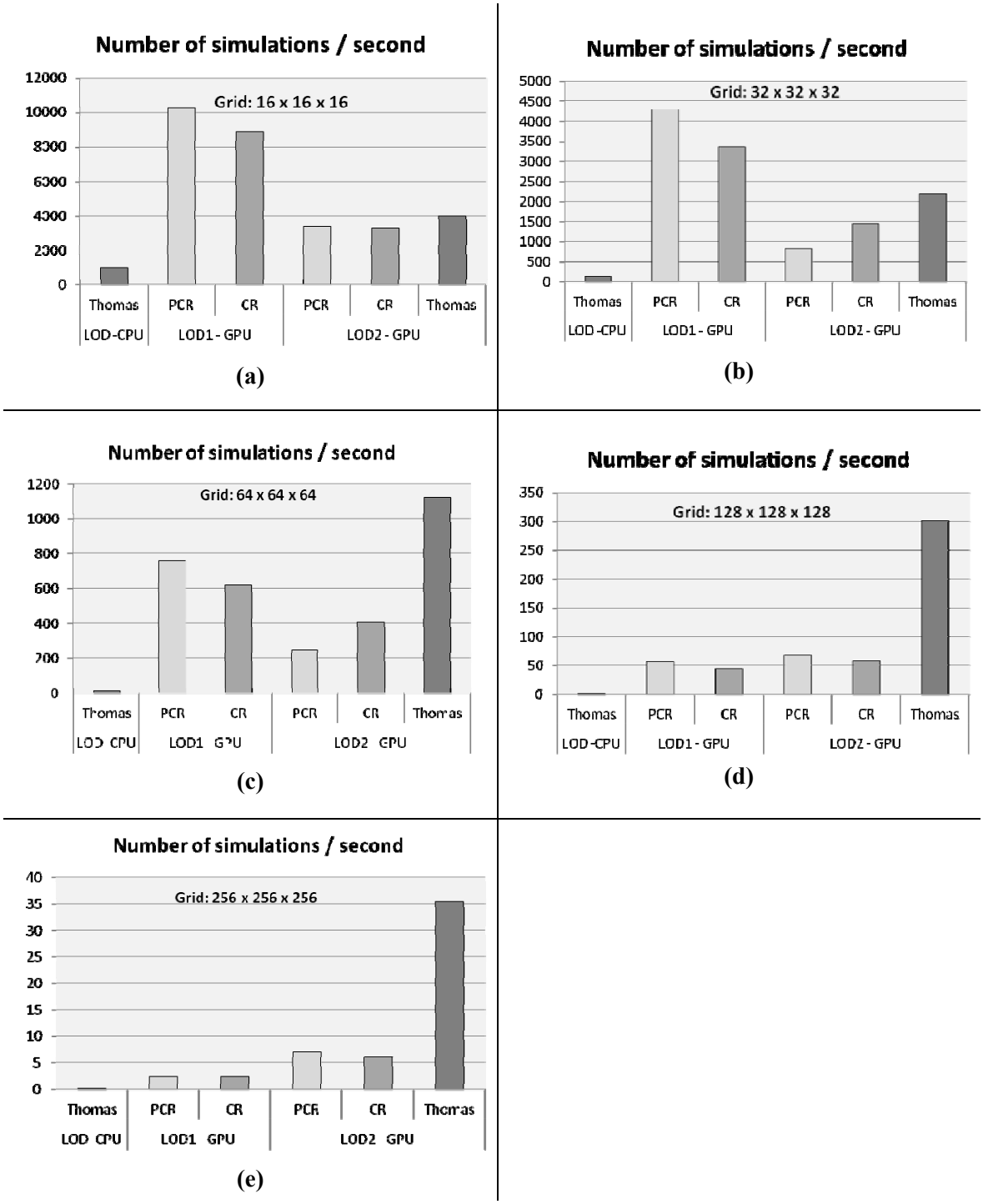


Figure 3 : Comparaison des rendements de solutions LOD sur différentes tailles de grilles

B.4.5 Implémentation hybride à haute performance de la LOD sur GPU

Comme déjà mentionné dans la section précédente, la figure 4 montre que le premier schéma (LOD1) de parallélisation avec solveur PCR est le plus performant pour les petits nombres de mailles et que le deuxième schéma (LOD2) de parallélisation employé avec le sovleur de Thomas devient le plus efficace à partir d'un certain seuil entre 48^3 et 64^3. En effet, le solveur de Gauss

met en jeu le plus petit nombre d'opérations par rapport aux autres solveurs. Néanmoins, en raison de son exécution séquentielle, il doit etre appliqué à un grand nombre de systèmes tridiagonaux simultanément afin de tirer parti des performances du GPU. C'est pourquoi, il surpasse les aurtres algorithmes lorsque le nombre de mailles est assez grand. En conséquence, le choix entre LOD1-PCR et LOD2-Gauss peut être fait en fonction du nombre de mailles dans la grille afin d'atteindre les meilleures performances.

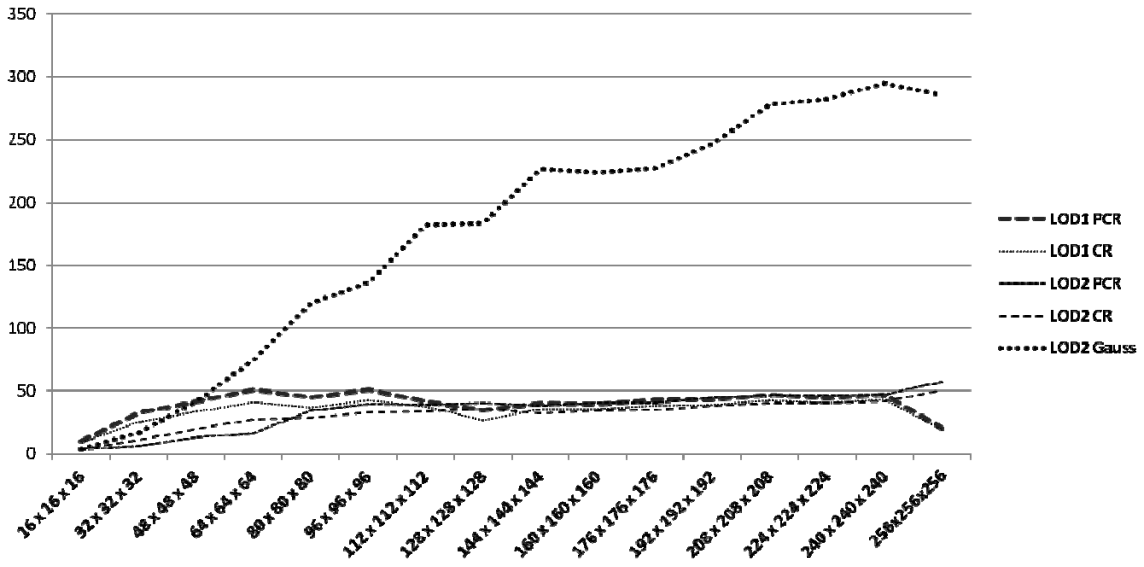


Figure 4 : Variation des rendemeents de solutions LOC en foncion de la taille dela grille

Considérons maintenant une grille avec un nombre de mailles $I \times J \times K$ selon les trois directions principales, et où au moins une des trois valeur de I, J ou K est différente des autres. Soit par exemple, une grille avec I fixé à 64 et $J \times K$ variable. Ensuite, le calcul est exécuté en considérant uniquement l'étape 1 de la LOD (Equation 7.a). Ceci équivaut à résoudre $J \times K$ systèmes tridiagonaux de taille $I \times I$. La solution est testée pour les deux solutions LOD1-PCR et LOD2-Gauss et est présentée figure 5. Le résultat est similaire au résultat de la figure 4, avec $J \times K$ assez grand (ici $J \times K > 3000$) LOD2-Thomas est plus performant que LOD1-PCR. Enfin, on conclut que pour chacune des trois étapes de calcul de la LOD (équations 7.a à 7.c), l'un des deux schémas LOD1-PCR ou LOD2-Thomas donne de meilleures performances, selon le produit du nombre de mailles dans les deux directions non-principales à cette étape. Le solveur hybride consiste ensuite à employer le schéma le plus efficace à chacune des trois étapes de la LOD selon le nombre de mailles dans les deux directions non principales à l'étape courante.

Cette approche est testée sur une grille de taille 48 × 256 × 32. La comparaison entre l'approche hyride et les schémas PCR-LOD1 LOD2-Thomas seuls est présentée figure 6. L'approche hybride emploie LOD1- PCR pour la solution de l'étape 2 de calcul et LOD2-Thomas pour par la solution des étapes 1 et 3. Il en résulte que le temps de calcul avec l'approche hybride est 85 fois plus rapide que le temps d'exécution sur CPU, bien que le nombre de mailles de la grille soit relativement faible, alors que LOD1-PCR et LOD2-Thomas seuls donnent respectivement des temps de calcul 50 et 25 fois plus rapides que le temps CPU.

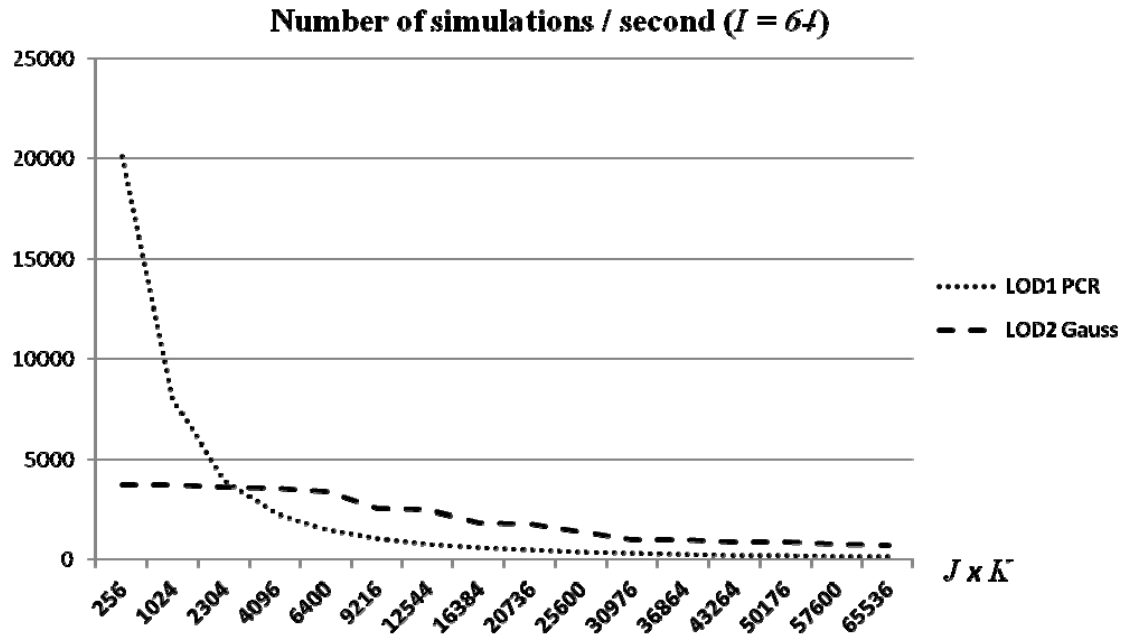


Figure 5 : Comparaison des deux meilleures solutions LOD pour différents $J \times k$ et I fixe

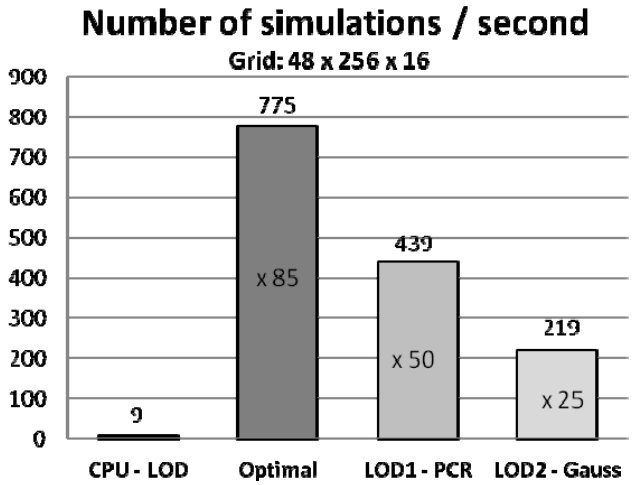


Figure 6 : Comparaison des rendements du schema optimal comparativement aux autres schémas pour une grille de dimensions $48 \times 256 \times 16$

B.4.6 Conclusions

Une implémentation efficace des méthodes des différences finies pour la résolution de l'équation différentielle de la diffusion de chaleur en 3D sur GPU a été présentée. Tout d'abord, la discrétisation en différences finies de l'équation de la diffusion de la chaleur tridimensionnelle a été rappelée. Ensuite, les méthodes explicite et implicite simple et le schéma de Crank -Nicolson ont été rapidement rappelés. Ces méthodes sont simples, néanmoins leurs solutions sont longues en temps de calcul. Comme solution, la méthode localement unidimensionnelle (LOD) a été adoptée.

La LOD permettant de résoudre l'équation de la diffusion en N^2 matrices tridiagonales de taille $N \times N$ comparativement à la résolution d'une matrice de taille $N^3 \times N^3$ pour la méthode implicite et la méthode de Crank-Nicolson.

L'algorithme de Thomas pour la résolution séquentielle des systeème d'équations linéaires tridiagonaux et les algorithmes de réduction cyclique et de réduction cyclique parallèle pour la résolution parallèle de ces systèmes ont ensuite été rappelés et comparés en fonction du nombre d'opérations et d'étapes de chacun. D'autre part, deux schémas de parallélisation de la LOD ont été présentés, la parallélisation maximale 3D et la parallélisation par plans. Le premier est nouveau et aboutit à un nombre de fils parallèles maximal et le second induit moins de parallélisation dans le but d'être applicable avec l'algorithme de Thomas. Ensuite, deux implémentations efficaces ont été retenues. La première est basée sur le premier schéma de parallélisation avec l'algorithme de réduction cyclique parallèle et le second est basé sur le deuxième schéma de parallélisation avec l'algorithme de Thomas. La première implémentation est plus efficace pour un nombre de mailles faible tandisque la deuxième est la plus efficace dès qu'un certain nombre de mailles est atteint. Finalement, un algorithme hybride a été développé, basé sur ces deux implémentations afin de tirer le maximum de performances du GPU. L'algorithme hybride permet d'avoir des accélerations considérables du temps de calcul par rapport au temps de calcul CPU, même pour des grilles à faibles nombres de mailles.

Références

1. M. N. Ozisik, Finite Difference Methods in Heat Transfer, New York: CRC, 1994.
2. G. W. Recktenwald, Finite-Difference Approximations to the Heat Equation, 2011.
3. J. Crank and P. Nicolson, A Practical Method for Numerical evaluation of Solution of Partial Differential Equations of the Heat Conduction Type, Proc. Camb. Phil. soc., vol. 43, pp. 50-67.
4. D. W. Peaceman and H. H. rachford, The Numerical Solution of Parabolic and Elliptic Differential Equations, J. Soc. Ind. Appl. Math., vol. 3, pp. 28-41, 1955.
5. P. L. T. Brian, A Finite-Difference Method of Higher-Order Accuracy for the solution of three-Dimensional Transient Heat Conduction Problems, AIChE J., vo. 7, pp. 367-370, 1961.
6. J. Douglas , Alternating Direction Method for Three Space Variables, Numer. Math., vol. 4, pp. 41-63, 1962.
7. J. Douglas JR and J. E. Gunn, A General Formulation of Alternating Direction Methods-Part I. Parabolic and Hyperbolic Problems, Numerische Mathematik, vol. 6, pp. 428-453, 1964.
8. B. K. Larkin, Some Stable Explicit Difference Approximations to the Diffusion Equation, Math. Comput., vol. 18, no. 86, pp. 196-202, 1964.
9. NVIDIA. CUDA Technology; http://www.nvidia.com/CUDA. 2007
10. NVIDIA.CUDA Programming Guide http:// developer.download.nvidia.com/compute/DevZone/docs/html/C/doc/CUDA_C_Programming_Guide.pdf. 2011.
11. R. W. Hockney, A Fast Direct Solution of Poisson's Equation using Fourier Analysis, Journal of the ACM, vol. 12, no. 1, pp. 95-113, 1965.
12. B. L. Buzbee, G. H. Golub and C. W. Nielson, On direct Methods for the Solution of Poisson's Equations, SIAM J. Numer. Anal, vol. 13, no. 1, pp. 54-70, 1976.

13. D. A. Bini and B. Meini,The Cyclic Reduction Algorithm: From Poisson Equation to Stochastic Processes and Beyond, Numerical Algorithms, vol. 51, no. 1, pp. 23-60, 2009.

14. R. W. Hockney and C. R. Jesshope, Parallel Computers, Adam Higler, Bristol, 1981.

15. D. Egloff, High Performance Finite Difference PDE Solvers on GPUs, Technical Report, QuantA lea Gmbh 2010.

16. G. D. Smith, Numerical Solution of Partial Differential Equations: Finite Difference methods, 2nd ed., Oxford University Press, London, 1978.

17. L. fox, Numerical Solution of Ordinary and Partial differential Equations, Addison-Wesley, Reading, Mass.

18. E. D'yanakov, Difference Schemes with Splitting Operators for Multidimensional Unsteady Problems, USRR Comput. Math., vol. 3, pp 581-607, 1963.

19. N. Yanenko, Convergence of the Method of Splitting for the Heat Conduction Equations with Variable Coefficients, USSR Comput. Math., vol. 3, pp 1094-1100, 1963.

B.5 Application à la Simulation Ultra-Rapide d'un Four de Réchauffage Sidérurgique

B.5.1 Description du four

Considérons un four de réchauffage sidérurgique de brames d'acier de dimensions internes $36.275\,m \times 6.8\,m \times 3.8\,m$. La figure 1 montre les éléments à l'intérieur du four. Ce four peut contenir en moyenne 45 brames d'acier circulant sur quatre rails. Les brames d'acier ont des longueurs différentes, mais une même hauteur de $0{,}2\,m$ et une largeur de $0{,}8\,m$ (figure 1).

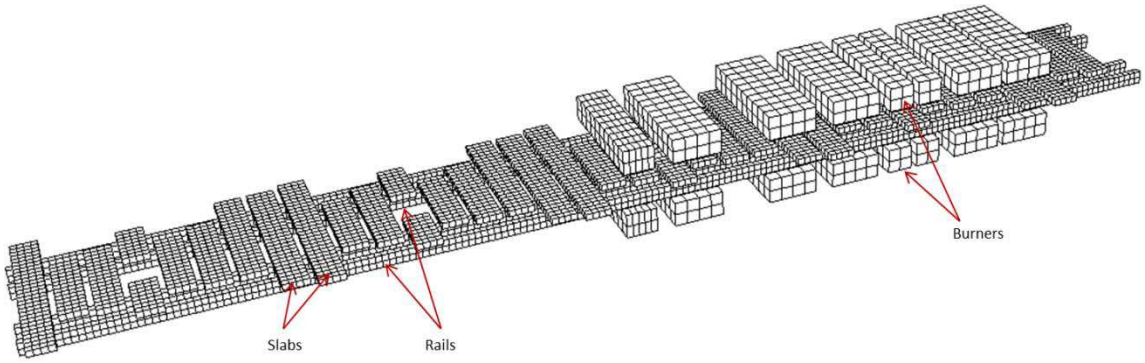


Figure 1 : Eléments à l'intérieur du four de réchauffage sidérurgique de brames d'acier

Une distance moyenne de $0{,}2\,m$ est supposée séparer les plaques d'acier sur toute la longueur du four. Les rails sont supposés avoir une section rectangulaire d'une taille de $0{,}2\,m \times 0{,}4\,m$. La première partie du four est une zone de préchauffage sans brûleur où les plaques reçoivent des flux de chaleur par recirculation des gaz chauds provenant de la zone de chauffage.

La zone de chauffage est équipée de brûleurs régéneratifs positionnés symétriquement dans les zones supérieure et inférieure du four. Tous les brûleurs sont identiques et les volumes de combustion sont supposées être de forme rectangulaire avec des dimensions de $0{,}8\,m \times 4\,m \times 0{,}8\,m$. Une émissivité de 0,9 est donnée aux parois du four. Les brames d'acier sont supposées avoir une émissivité de 0,8 et les rails une émissivité de 0,75. Le volume du four est rempli de gaz de combustion modélisés comme gris et diffus avec un coefficient d'absorption de 0,5 m^{-1}.

B.5.2 Calcul des facteurs d'échanges radiatifs directs

Les facteurs d'échanges radiatifs directs sont calculés par la méthode zonale à coefficients d'absorption multiples en utilisant l'approche multi- grille présentée dans le chapitre 2. L'approche multi- grille implique que chaque catégorie de facteurs d'échange est calculée avec une taille de maille appropriée afin de garantir une bonne précision tout en évitant tout temps de calcul non nécessaire. Les facteurs d'échanges directs entre brûleurs et entre parois du four sont calculés avec des mailles de 40 cm^3 (0,15 Mvoxels). Les facteurs d'échanges directs entre les murs et les brames d'acier, les rails et les brûleurs ainsi que entre les brûleurs et les murs sont calculés en utilisant un maillage de 20 cm^3 (1,2 Mvoxels). Enfin, un maillage de 10 cm^3 (plus de 9 millions de voxels) est

appliqué au four pour le calcul des facteurs d'échanges directs entre les dalles elles-mêmes et entre les dalles et les rails. A ce stade, un temps de calcul CPU de 1 200 secondes est atteint pour un calcul des facteurs d'échanges directs, tandis que l'exécution sur GPU est environ 320 fois plus rapide avec un temps de calcul de 3,6 secondes (CPU E5507 Intel Xeon à 2,27 GHz et GPU NVIDIA Tesla GPU C 1060).

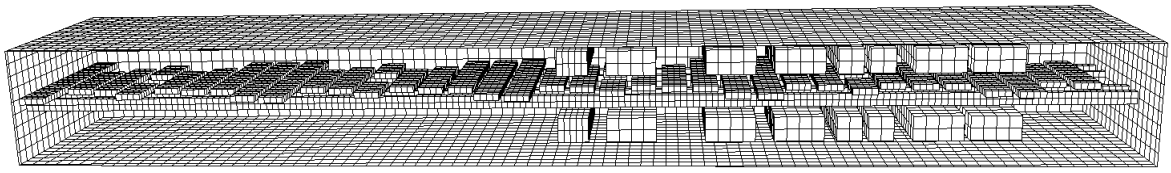


Figure 2 : Maillage résultant du calcul des facteurs d'échanges directs

A la fin, les facteurs d'échanges directs calculés sont associés ensemble afin de donner à chaque objet la plus grande taille de mailles qui lui a été appliquée au cours des calculs. La figure 2 montre le maillage final du four. La taille finale de maille est de 20 cm^3 pour les brames d'acier et les rails et de 40 cm^3 pour les brûleurs, le volume du four et les murs.

En revanche, un gain en temps de calcul peut être effectué en éliminant le calcul des facteurs d'échanges directs insignifiants. En effet, en raison du faible rapport entre la largeur et la hauteur du four sur sa longueur, le rayonnement provenant d'un élément de surface au milieu du four par exemple, est absorbé longtemps avant d'atteindre le début ou la fin du four (Figure 3). La distance limite à partir de laquelle l'échange radiatif devient négligeable est déterminée dynamiquement lors du calcul des facteurs d'échanges radiatifs. Le calcul se fait tout d'abord aux plans verticaux voisins à la maille de départ et s'éloigne progressivement jusqu'à ce qu'il soit arrêté lorsque la somme des facteurs d'échanges directs est égale à 99 % de sa valeur. Par conséquent, le temps de calcul CPU est réduit à 180 secondes et le temps GPU à moins d'une seconde.

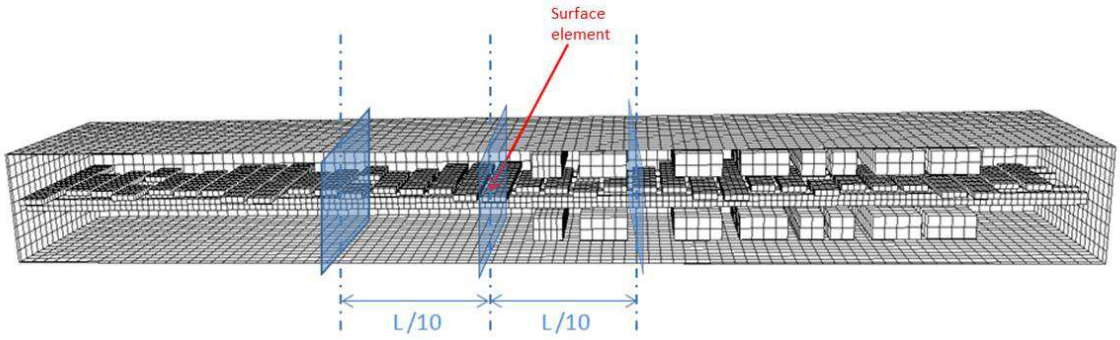


Figure 3 : Limites du calcul des facteurs d'échanges directs sur la longueur du four

B.5.3 Calcul des facteurs d'échanges radiatifs totaux

Les facteurs d'échanges totaux sont calculés à partir des facteurs d'échanges directs à l'aide de l'algorithme des revêtements. Le nombre total d'éléments de surface dans le four avec l'approche multi-grille est de 14 660. L'application de l'algorithme direct des revêtements au four avec ce nombre d'éléments de surfaces nécessiterait environ deux heures de calcul CPU.

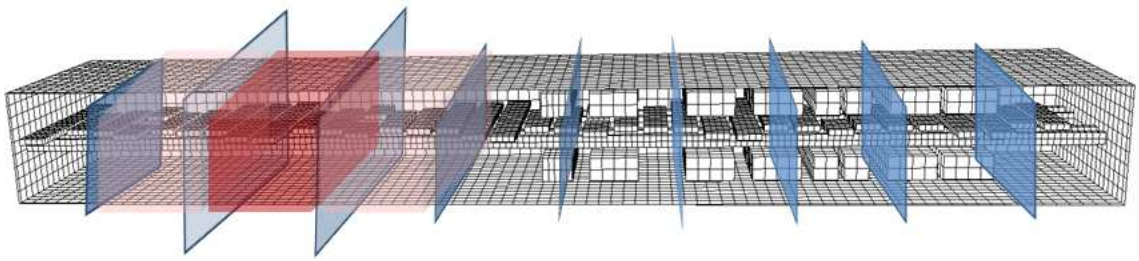


Figure 4 : Calcul des facteurs d'échanges totaux par zone

D'autre part, l'application directe du NRPA pour le calcul des facteurs d'échanges totaux nécessite un temps de calcul GPU de 9 secondes à l'ordre 2 et de 17 secondes à l'ordre 3. Les erreurs moyennes pondérées qui en résultent sont de 3 % et 1 % respectivement. Le temps de calcul avec le NRPA reste un ordre de grandeur plus élevé que le temps de calcul des facteurs d'échanges directs. En conséquence, il rest le facteur limitant pour le temps de calcul. En solution, les facteurs d'échanges totaux peuvent être estimés en divisant le four en un certain nombre de zones de calcul. L'application de la NRPA est ainsi effectuée sur plusieurs zones mais avec un nombre de surfaces inférieur, ce qui donnera des temps de calcul plus courts vu que le NRPA possède une complexité de l'ordre $O(n^{2.38})$. Le calcul par zone est représenté figure 4. Le four est tout d'abord divisé en N zones dans le sens de la longueur. Les facteurs d'échanges totaux d'une zone I sont ensuite calculés en tenant compte des multi-reflections dans les zones $I-1$, I et $I+1$. Pour $N = 10$, un temps de calcul de 0,75 secondes est réalisé pour le NRPA à l'ordre 3, NRPA avec une erreur supplémentaire variant entre 1 % et 2 %.

B.5.4 Calcul des profils de température dans les brames d'acier

Le profil de température dans les brames est donné par le calcul de la diffusion de la chaleur dans les brames avec les conditions limites de flux radiatif calculées. La diffusion de la chaleur dans les brames est calculée par la méthode LOD (chapitre 4). Un maillage fin est appliqué aux plaques pour le calcul.

Une discrétisation centrale du premier ordre en différences finies est considérée pour les conditions aux limites de flux afin de garantir que les résultats soient précis au second ordre par rapport au temps.

Le calcul est effectué à l'aide de l'implémentation GPU hybride présentée au chapitre précédent. Des temps de calcul très rapides sont réalisés. Par exemple, avec une grille de $32 \times 64 \times 16$ mailles, un temps de calcul inférieur à 1/3000 secondes est obtenu pour le calcul relatif à un pas de temps. Un profil de température 3D très précis dans toutes les brames est finalement obtenu.

B.5.5 Calcul des facteurs d'échanges radiatifs totaux

Les brames se déplacent dans le four à une vitesse de $0{,}0025\ ms^{-1}$. Une meilleure précision est obtenue pour les calculs dynamiques si le calcul des facteurs d'échanges radiatifs est effectué surr le plus petit pas d'espace possible dans la direction de déplacement des brames. D'après la section précédente, la plus grande taille de maille utilisée pour le calcul des facteurs d'échanges directs entre un élément de surface de brame et un autre élément de surface dans le four est de $20\ cm$. Cela implique que le plus petit pas d'espace qui pourra être considéré pour le calcul de facteurs d'échanges radiatifs est de $20\ cm$. Les facteurs d'échanges radiatifs doivent ensuite être calculés pour un pas de distance de $20\ cm$ ou également un pas de temps de 80 secondes. Le temps de calcul qui est nécessaire pour réaliser le calcul des facteurs d'échanges radiatifs par rapport à l'intervalle de temps est d'environ $1{,}75$ secondes (NVIDIA Tesla GPU C 1060). Un intervalle de temps de 80 secondes est alors suffisamment grand pour permettre la simulation dynamique en temps réel du four, parce qu'il est largement supérieur à $1{,}75$ secondes. En conséquence, la diffusion de la chaleur devra être calculée avec un pas de temps inférieur à 80 secondes. Même si les facteurs d'échanges radiatifs sont constants sur une période de 80 secondes de temps, le flux de chaleur par rayonnement sur les brames varie au cours de cette période, étant donné que les températures des brames et du four varient en continu.

Une grille de $32 \times 64 \times 16$ mailles pour le calcul de la diffusion de la chaleur dans les brames permet de garder une bonne précision tout en garantissant un temps de calcul rapide. Le temps de calcul correspondant avec ce maillage est de $1/3000$ seconde par brame et par pas de temps c'est-à-dire. que le calcul de la diffusion dans les 45 surfaces de four, necessite $0{,}015$ secondes par pas de temps. De toute évidence, des pas de temps beaucoup plus élevés seront considérés pour le calcul de la diffusion de la chaleur, étant donné que la précision des résultats ne sera pas affectée de façon significative.

En résumé, la simulation dynamique en temps réel est effectuée avec un pas de temps de 80 secondes pour le calcul des facteurs d'échanges radiatifs et un pas de temps, pour le calcul de la diffusion, supérieur à 0,015s (NVIDIA Tesla GPU C 1060).

En outre, le contrôle de la ligne de chauffe appliqué doit être mis à jour toutes les 10 secondes, en tenant compte du fait qu'un pas de temps de 80 secondes est imposé pour les facteurs d'échanges radiatifs et un pas de temps de 5 secondes pour le calcul de la diffusion de la chaleur. Le temps de calcul qui est nécessaire pour effectuer un mouvement virtuel des surfaces de 20 cm sera alors 1,75 secondes pour le calcul des facteurs d'échanges radiatifs et de 0,24 secondes pour le calcul de la diffusion de la chaleur en utilisant un NVIDIA Tesla GPU C 1060, soit environ 2 secondes. Ceci équivaut à 2 secondes de temps de calcul pour un déplacement des brames d'une distance de 20 cm. Par conséquent, un calcul prédictif pour les brames se déplaçant d'une distance de 1 m peut être effectué dans la période de 10 secondes précédant chaque mise à jour des entrées du four. Cette performance est directement liée à la performance du processeur graphique. Compte tenu qu'un GPU plus récent, le NVIDIA GeForce GTX 690 par exemple, possède deux GPU avec 3 072 cœurs de calcul au total, tandis que le C 1060 en a seulement 240, le GTX aboutira facilement à des temps de calcul cinq fois plus courts que le C 1060. En conséquence, si une station informatique possède

deux NVIDIA GeForce GTX 690 GPU, un calcul prédictif sur 20 mètres sur le mouvement des plaques pourrait être effectué en continu. Ce débit de calcul aurait nécessité un cluster de CPU d'environ un millier de processeurs.

Modélisation Ultra-rapide des Transferts de Chaleur par Rayonnement et par Conduction et exemple d'application

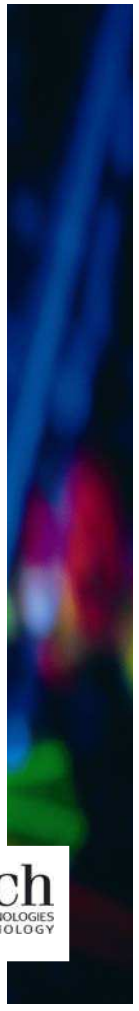
RESUME : L'apparition de CUDA en 2007 a rendu les GPU hautement programmables permettant ainsi aux applications scientifiques et techniques de profiter de leur capacité de calcul élevée. Des solutions ultra-rapides pour la résolution des transferts de chaleur par rayonnement et par conduction sur GPU sont présentées dans ce travail. Tout d'abord, la méthode MACZM pour le calcul des facteurs de transferts radiatifs directs en 3D et en milieu semi-transparent est représentée et validée. Ensuite, une implémentation efficace de la méthode à la base d'algorithmes de géométrie discrète et d'une parallélisation optimisée sur GPU dans CUDA atteignant une accélération de 300 à 600 fois, est présentée. Ceci est suivi par la formulation du NRPA, une version non-récursive de l'algorithme des revêtements pour le calcul des facteurs d'échanges radiatifs totaux. La complexité du NRPA est inférieure à celle du PA et son exécution sur GPU est jusqu'à 750 fois plus rapide que l'exécution du PA sur CPU. D'autre part, une implémentation efficace de la LOD sur GPU est présentée, consistant d'une alternance optimisée des solveurs et schémas de parallélisation et achevant une accélération GPU de 75 à 250 fois. Finalement, toutes les méthodes sont appliquées ensemble pour la résolution des transferts de chaleur en 3D dans un four de réchauffage sidérurgique de brames d'acier. Dans ce but, MACZM est appliquée avec un maillage multi-grille et le NRPA est appliqué au four en le découpant en zones, permettant d'obtenir un temps de calcul très rapide avec une précision élevée. Ceci rend les méthodes utilisées de très grande importance pour la conception de stratégies de contrôle efficaces et précises.

Mots clés : Méthode Zonale à Multiples Coefficients d'Absorption (MACZM), Algorithme des revêtements (PA), Algorithme des revêtements non-récursif (NRPA), Méthode des différences finies Localement unidimensionnelle (LOD), Processeurs Graphiques (GPU), CUDA (architecture de dispositif de calcul unifié).

Fast Modeling of Radiation and Conduction Heat Transfer and application example

ABSTRACT: The release of CUDA by NVIDIA in 2007 has tremendously increased GPU programmability, thus allowing scientific and engineering applications to take advantage of the high GPU compute capability. In this work, we present ultra-fast solutions for radiation and diffusion heat transfer on the GPU. First, the Multiple Absorption Coefficient Zonal Method (MACZM) for computing direct radiative exchange factors in 3D semi-transparent media is reviewed and validated. Then, an efficient implementation of MACZM is presented, based on discrete geometry algorithms, and an optimized GPU CUDA parallelization. The CUDA implementation achieves 300 to 600 times speed-up. The Non-recursive Plating Algorithm (NRPA), a non-recursive version of the plating algorithm for computing total exchange factors is then formulated. Due to low-complexity matrix multiplication algorithms, the NRPA has lower complexity than the PA does and it runs up to 750 times faster on the GPU by comparison to the CPU PA. On the other hand, an efficient GPU implementation for the Locally One Dimensional (LOD) finite difference split method for solving heat diffusion is presented, based on an optimized alternation between parallelization schemes and equation solvers, achieving accelerations from 75 to 250 times. Finally, all the methods are applied together for solving 3D heat transfers in a steel reheating furnace. A multi-grid approach is applied for MACZM and a zone-by-zone computation for the NRPA. As a result, high precision and very fast computation time are achieved, making the methods of high interest for building precise and efficient control units.

Keywords : Multiple Absorption Coefficient Zonal Method (MACZM), Plating algorithm (PA), Non-Recursive Plating Algorithm (NRPA), Locally One Dimensional (LOD) finite difference method, Graphics Processing Unit (GPU), Compute Unified Device Architecture (CUDA).



MINES ParisTech

ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY